Scalable Reinforcement Learning for Linear-Quadratic Control of Networks

Johan Olsson

LUND UNIVERSITY

Department of Automatic Control

MSc Thesis
TFRT-6207
ISSN 0280-5316

Department of Automatic Control
Lund University
Box 118
SE-221 00 LUND
Sweden

Printed in Sweden by Tryckeriet i E-huset
Lund 2023

Abstract

Distributed optimal control is known to be challenging and can become intractable even for linear-quadratic regulator problems. In this work, we study a special class of such problems where distributed state feedback controllers can give near-optimal performance. More specifically, we consider networked linear-quadratic controllers with decoupled costs and spatially exponentially decaying dynamics. We aim to exploit the structure in the problem to design a scalable reinforcement learning algorithm for learning a distributed controller. Recent work has shown that the optimal controller can be well approximated only using information from a κ-neighbourhood of each agent. Motivated by these results, we show that similar results hold for the agents' individual value and action-value functions. We continue by designing an algorithm, based on the actor-critic framework, to learn distributed controllers only using local information. Specifically, the action-value function is estimated by modifying the Least Squares Temporal Difference for Q-functions method to only use local information. The algorithm then updates the policy using gradient descent. Finally, the algorithm is evaluated through simulations which suggest near-optimal performance.

Acknowledgements

This thesis was completed in the summer of 2023 as part of my Master's Degree in Engineering Physics from The Faculty of Engineering at Lund University (LTH). Most of the work was carried out at Harvard University where I was a visiting researcher between January and June 2023.

Several people have contributed to this work, and without them, this thesis would not have been possible. First and foremost, I would like to thank Professor Na Li at Harvard University for inviting me to visit her group and for advising and guiding me while I was working on this thesis. Professor Na Li's students also deserve a big thank you for the warm welcome. Especially Runyu Zhang, for the endless discussions and input on my work. I would also like to thank Emma Tegling at the Department of Automatic Control at Lund Univeristy for the continuous support and ideas on how to enhance this thesis. Finally, I would like to extend my appreciation to my examiner Anders Rantzer at the Department of Automatic Control at Lund Univeristy for reviewing the thesis and for making this project possible by introducing me to Professor Na Li.

Contents

1. Introduction **9**
- 1.1 Motivation . . . 9
- 1.2 Related Work . . . 10
- 1.3 Contribution . . . 10
- 1.4 Outline . . . 10

2. Background **12**
- 2.1 The Linear-Quadratic Regulator . . . 12
- 2.2 Continuous Reinforcement Learning for Average Cost Problems 13
- 2.3 Learning and Control of Multi-Agent Systems . . . 16
- 2.4 Individual Value and Action-Value Functions . . . 17

3. Problem Formulation **19**
- 3.1 Problem Setup . . . 19
- 3.2 Spatially Exponentially Decaying Structure . . . 21
- 3.3 Stable Systems and Stabilising Controllers . . . 24
- 3.4 Structure of the Optimal Controller . . . 24
- 3.5 Problem Statement . . . 25

4. Individual Value and Action-Value Functions for Network LQR **26**
- 4.1 Derivation of the Individual Value and Action-Value Functions in the Network LQR setting . . . 26
- 4.2 Spatially Decaying Structure of the Individual Value and Action-Value Functions . . . 27

5. Scalable MARL for Network LQR **33**
- 5.1 Least-Squares Temporal Difference Learning for Centralised Q-Function Evaluation . . . 33
- 5.2 Critic Architecture: Decentralised Q-function estimation . . . 34
- 5.3 Actor Architecture: Decentralised Policy Update . . . 38
- 5.4 Scalable MARL for Network LQR . . . 41

6. Simulation Study **42**
- 6.1 Lemma 4.1 for a Toy Problem . . . 42

6.2 Scalable MARL for Thermal Control 44
6.3 Results and Discussion 46
7. Conclusions and Future Work **51**
7.1 Conclusions 51
7.2 Limitations and Directions for Future Work 51
Bibliography **53**

1

Introduction

1.1 Motivation

Multi-agent networked systems such as power grids, wireless communications networks and smart buildings have been studied extensively, both from the control community and the machine learning community. Due to the scale of these systems centralised control is often considered unfeasible and instead algorithms that distribute the control over the agents in the network are needed [2]. These controllers do not necessarily depend on the global state, but instead control the system using information from the agent itself and its neighbours. The distributed control problem is, however, known to be difficult in general. In fact, a classic result in distributed control show that even in the setting of linear dynamics with Gaussian noise and quadratic costs (LQR), synthesis of the optimal distributed controller is a challenging task [28].

Similarly for multi-agent reinforcement learning (MARL), the curse of dimensionality quickly becomes a problem [20]. Ignoring the issue of scalability, the success of reinforcement learning (RL) in a wide variety of applications such as games [24, 27], robotics [11] and autonomous driving [16] makes this approach interesting for control synthesis in networked systems. Taking scalability into account, recent work has shown promising results when applying reinforcement learning to networked systems with local information structure [20].

In this work, we aim to study how the structure of the problem can be utilised to develop a scalable algorithm for learning to control networked systems. More specifically we are considering a network of linear-quadratic (LQ) controllers with a spatially exponentially decaying (SED) structure between nodes. Spatial exponential decay is a system property that is connected to the underlying graph's topology and is formally defined in Definition 3.2. Intuitively a system is SED if the dependence between agents in the network decays exponentially with the distance between them. Recent results on the structure of the optimal controller tell us distributed control of such systems is feasible [30]. Furthermore, for the single agent case, the authors of [14] provide performance guarantees when applying centralised reinforcement learning to synthesise LQ controllers. The goal is to combine these

results and design a scalable learning algorithm for distributed control of networked LQ agents.

1.2 Related Work

In the multi-agent case, reinforcement learning algorithms for networked agents with information decay has been previously studied in [19, 20, 32]. In these works reinforcement learning algorithms for multi-agent systems are studied in a more general context and not specifically from a LQ perspective. In short, they show that there are multi-agent systems with localised interactions that allow for distributed learning.

More similar to our work is that in [17] where they study distributed reinforcement learning for multi-agent LQ control. They assume each agent observes a partition of the global state and study synthesis of local linear state feedback policies of the partial observations. They assume the agents can communicate through some predetermined communication network and rely on consensus and derivative-free optimisation to find a distributed controller.

Another work similar to ours is that in [9] where a distributed Q-learning procedure is proposed for the linear-quadratic control problem. The algorithm is evaluated through a simulation study indicating near-optimal results. However, analysis of the algorithm is left to future work.

1.3 Contribution

The contribution of this work is twofold. Firstly, the structure of the individual agents' value functions and action-value functions is studied. This analysis culminate in Theorem 4.3 which shows how the system's spatially decaying structure is preserved for the agents' individual value functions. Corollary 4.4 then extends this result to the individual action-value functions.

Secondly, an algorithm for distributed learning and control for network LQR is proposed. The design is motivated by the structure of the individual action-value functions and based on the actor-critic framework. Finally, the proposed algorithm is analysed through numerical experiments.

1.4 Outline

- **Chapter 2 – Background:** Brief overview of linear-quadratic control, reinforcement learning and multi-agent systems.
- **Chapter 3 – Problem Formulation:** Introduces the specific model and assumptions. The spatially exponentially decaying property is defined and our problem formally stated.

- **Chapter 4 – Individual Value and Action-Value Functions for Network LQR:** The rate of decay for the individual value and action-value functions is studied and the error of truncating these functions is bounded.
- **Chapter 5 – Scalable MARL for Network LQR:** Motivated by the results of the previous section, this part concerns algorithm design for scalable control for the network LQR problem.
- **Chapter 6 – Simulation Study:** The results from the two previous sections are studied using numerical simulations and the results are discussed.
- **Chapter 7 – Conclusions and Future Work:** Summary and concluding remarks on this work followed by a discussion on limitations and possible future directions.

2

Background

This chapter provides necessary background information and notation that will be used throughout the rest of this work. First, the classical LQR problem and its optimal solution is introduced. Then, a brief background on concepts in single agent continuous reinforcement learning is provided. Finally, the LQR problem and continuous reinforcement learning are discussed when the system consists of multiple agents.

2.1 The Linear-Quadratic Regulator

The infinite horizon discrete time LQR problem is a classic problem in optimal control [4]. In a discrete time setting, the state at time t is governed by the linear equation

$$x(t+1) = Ax(t) + Bu(t) + w(t), \quad w(t) \sim \mathcal{N}(0, \sigma_w^2 I)$$

where $x \in \mathbb{R}^n$ is the state, $u \in \mathbb{R}^m$ the input and $w \in \mathbb{R}^n$ system noise. It is assumed that A and B are controllable, meaning the system can transition from any state to any other state in a finite time [12]. At each timestep there is also a cost incurred by

$$c(t) = x(t)^\top Sx(t) + u(t)^\top Ru(t)$$

where $S \in \mathbb{R}^{n \times n}$ and $R \in \mathbb{R}^{m \times m}$ are positive semidefinite cost matrices. The goal is to minimise the average cost over an infinite horizon by choosing the input u. Mathematically formulated the LQR problem aims to solve the optimisation problem:

$$\begin{aligned} \min_{\{u_t\}_{t=0}^{\infty}} \; & J(u) := \lim_{T \to \infty} \mathbb{E}\left[\frac{1}{T} \sum_{t=0}^{T-1} \Big(x(t)^\top Sx(t) + u(t)^\top Ru(t) \Big) \right] \\ s.t. \; & x(t+1) = Ax(t) + Bu(t) + w(t), \quad w(t) \sim \mathcal{N}(0, \sigma_w^2 I) \end{aligned}$$

It is well known that the solution to the LQR problem is given by the linear state feedback controller $u^* = K^* x$ where

$$K^* = -(R + B^\top P^* B)^{-1} B^\top P^* A \tag{2.1}$$

This means that given any state x the optimal action is given by $u^* = K^* x$. In Equation (2.1), P^* is the solution to the Discrete Algebraic Riccati Equation [4]

$$P^* = A^\top P^* A - A^\top P^* B(R + B^\top P^* B)^{-1} B^\top P^* A + S$$

and by [3], the optimal cost incurred by following K^* is given by

$$\min_u J(u) = J(K^*) = \operatorname{tr}(P^*)$$

When the dynamics are unknown, the optimal controller can obtained by collecting data and then either model the system or find a controller directly from the data. A general framework for learning from data is reinforcement learning which will be considered next.

2.2 Continuous Reinforcement Learning for Average Cost Problems

The aim of this section is to introduce the most important concepts from reinforcement learning and the interested reader is referred to a textbook on the subject, for example Sutton and Bartos book *Reinforcement learning: An introduction* from 2018 [25].

Reinforcement learning is a machine learning paradigm where one or more agents learn by interacting with the environment. The interaction between agent and environment is commonly modelled as a Markov decision process (MDP) which is a tuple $(\mathscr{X}, \mathscr{U}, c, P)$. Let $\Delta_{\mathscr{X}}$ denote the distribution over states, then

$$\begin{aligned}
&\mathscr{X} \subset \mathbb{R}^n \text{ is the state space,}\\
&\mathscr{U} \subset \mathbb{R}^m \text{ is the action space,}\\
&c : \mathscr{X} \times \mathscr{U} \to \mathbb{R} \text{ is a cost function,}\\
&P : \mathscr{X} \times \mathscr{U} \to \Delta_{\mathscr{X}} \text{ is the transition probability}
\end{aligned}$$

For each timestep t, the agent receives state $x(t) \in \mathscr{X}$ from the environment and chooses an action $u(t) \in \mathscr{U}$ which lead to a penalty $c(t) = c(x(t), u(t))$. This action also leads to the environment transitioning from $x(t)$ to $x(t+1)$ via $x(t+1) \sim P(x(t), u(t))$. In reinforcement learning, the transition probability is assumed to be unknown and the goal is to learn a policy which is a mapping $\pi : \mathscr{X} \to \mathscr{U}$ from the current state to an action or distribution over actions. If the agent follows a policy π

it will take action $\pi(x)$ upon receiving state x from the environment. A visualisation of how the agent interact with the environment can be seen in Figure 2.1.

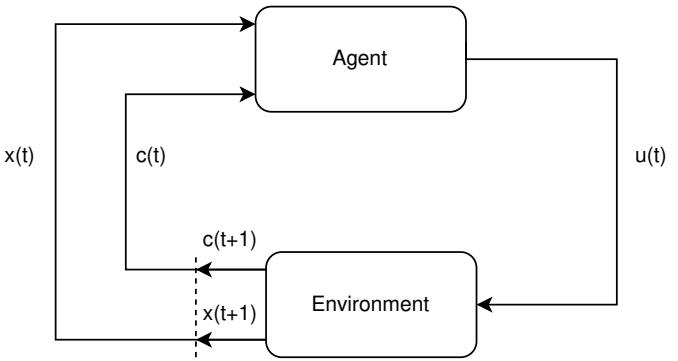


Figure 2.1 General schema visualising how a reinforcement learning agent interact with the environment.

For a given policy π we can define the value function and action-value function and use these to find a better policy. The value function gives the cost of following π from state x and is given by

$$V^{\pi}(x) := \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty}(c(x(t),\pi(x(t))) - \lambda^{\pi}) \mid x(0) = x\right] \tag{2.2}$$

where λ^{π} is the expected average stage cost of policy π under stationarity

$$\lambda^{\pi} := \lim_{T\to\infty} \mathbb{E}_{\pi}\left[\frac{1}{T}\sum_{t=0}^{T} c(x(t),\pi(x(t)))\right]$$

and similarly the cost of taking an arbitrary action u from state x and thereafter following π is given by the action-value function

$$Q^{\pi}(x,u) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty}(c(x(t),u(t)) - \lambda^{\pi}) \mid x(0) = x, u(0) = u\right] \tag{2.3}$$

We note that we can separate out the first term of the summation in Equation (2.2) and (2.3) and thus see that the Bellman equations

$$V^{\pi}(x) = c(x,\pi(x)) - \lambda^{\pi} + \mathbb{E}_{x'\sim P(x,\pi(x))}[V^{\pi}(x')], \tag{2.4}$$

$$Q^{\pi}(x,u) = c(x,u) - \lambda^{\pi} + \mathbb{E}_{x'\sim P(x,\pi(x))}[Q^{\pi}(x',\pi(x'))] \tag{2.5}$$

hold for any state $x \in \mathscr{X}$ and action $u \in \mathscr{U}$ for both the value and the action-value functions. Moreover, there is an apparent similarity between the value function and

the action-value function and we note that by setting $u = \pi(x)$ in the definition of $Q^\pi(x,u)$ we recover the value function. That is

$$Q^\pi(x,\pi(x)) = V^\pi(x)$$

Similarly, an important result in reinforcement learning is the policy improvement theorem which says that if two deterministic policies π and π' are such that

$$Q^\pi(x,\pi'(x)) \leq V^\pi(x)$$

then π' must be at least as good as π.

The policy improvement theorem has led to many different approaches to learning the policy π and it is impossible to go through them all, instead we briefly introduce a class of reinforcement learning algorithms known as actor-critic methods which will be used in this work.

In actor-critic methods there is an actor which selects the actions and there is a critic which evaluates the policy generated by the actor. The actor-critic is a general framework and how one chooses the actor and the critic typically depends on the problem. One way to construct this architecture is to let the critic construct an approximate action-value function using Equation (2.5) which is then used by the actor to update the policy parameters in a direction of improvement [13].

We end this section by specifying the value and action-value functions in the context of LQR. For the LQR problem, the value function and action-value function for any stable linear policy $\pi(x) = Kx$ can be found using equations (2.4) and (2.5). Using a quadratic ansatz, it can be shown that the value function $V^K(x)$ and the action-value function $Q^K(x,u)$ are both quadratic [1]. For readability, we drop the policy superscript when talking about LQR and remember that, for example $V = V^\pi = V^K$ depend on the policy $\pi(x) = Kx$. For a stable linear policy K, the value and action-value functions for the LQR problem satisfy

$$V(x) = x^\top P x \tag{2.6}$$

$$Q(x,u) = \begin{pmatrix} x^\top & u^\top \end{pmatrix} H \begin{pmatrix} x \\ u \end{pmatrix} \tag{2.7}$$

where P is the solution to the Lyapunov equation

$$P = S + K^\top R K + (A + BK)^\top P (A + BK) \tag{2.8}$$

and

$$H = \begin{pmatrix} H_{11} & H_{12} \\ H_{12}^\top & H_{22} \end{pmatrix} = \begin{pmatrix} S + A^\top P A & A^\top P B \\ B^\top P A & R + B^\top P B \end{pmatrix} \tag{2.9}$$

Moreover, in [4] it is shown that the average cost of following K is given by

$$\lambda = \operatorname{tr}\left(\sigma_w^2 P\right) = \operatorname{tr}\left(\sigma_w^2 H \begin{pmatrix} I \\ K \end{pmatrix} \begin{pmatrix} I \\ K \end{pmatrix}^\top\right) \tag{2.10}$$

2.3 Learning and Control of Multi-Agent Systems

In this work we will consider multi agent systems where the agents' dynamics are coupled and the coupling is due to the agents being interconnected in a network structure modelled by a graph. This setting comes with several new challenges, both for controlling and learning the system.

In centralised control, all agents send information about their state to a central computing unit which then uses the global state information to decide the next control action. When the system consists of many agents, centralised control can become unfeasible due to memory and computational requirements which might mean a distributed control architecture is the only option [22]. Distributed control differs from centralised control in the sense that agents share information with only a few other agents, or not at all. The agents then decide the next action using only the information they received from other agents [22]. Distributed control requires less memory and computation but can, of course, never achieve a lower cost than centralised control. Besides computational aspects, distributed control is appealing because of its robust nature. If a single subsystem fails, global system failure can still often be avoided [6].

In the reinforcement learning literature, the study of systems with several agents is known as Multi-Agent Reinforcement Learning (MARL). In MARL the global action and state space are given by the product of the individual spaces [6]. For a system consisting of N agents where the state and action space of Agent i is given by $\mathscr{X}_i$ and $\mathscr{U}_i$ respectively, the global state space $\mathscr{X}$ and action space $\mathscr{U}$ are given by the product spaces.

$$\begin{aligned} \mathscr{X} &:= \mathscr{X}_1 \times \mathscr{X}_2 \times \cdots \times \mathscr{X}_N \\ \mathscr{U} &:= \mathscr{U}_1 \times \mathscr{U}_2 \times \cdots \times \mathscr{U}_N \end{aligned}$$

The dimensions of the state and action spaces thus grow linearly with the size of the system and, just as in the control case, one wants to lower the computational and storage costs by distributing the learning across the agents [6].

In general, optimal decentralised control is NP-hard and as far as we know additional conditions or structure on the problem is needed in order for efficient algorithms to even exist for the problem [8]. Due to the difficulty of the problem, suboptimal algorithms are interesting and one can then study the trade-off between optimality and scalability [20]. These algorithms can, similarly to the single agent case, be based on learning value and action-value functions which we now introduce in the MARL setting.

2.4 Individual Value and Action-Value Functions

In the MARL setting, we introduce the concept of individual value and action-value functions. For problems where the global cost is defined as the sum of the individual costs,

$$c(t) := \sum_{i=1}^{N} c_i(t) \tag{2.11}$$

there is a natural way to define these individual functions. Inspired by [19], we first define the expected average stage cost for the individual costs as

$$\lambda_i^{\pi} := \lim_{T\to\infty} \mathbb{E}_{\pi}\left[\frac{1}{T}\sum_{t=0}^{T} c_i(x(t),\pi(x(t)))\right]$$

and note that

$$\begin{aligned}\lambda^{\pi} &= \lim_{T\to\infty} \mathbb{E}_{\pi}\left[\frac{1}{T}\sum_{t=0}^{T} c(x(t),\pi(x(t)))\right] = \lim_{T\to\infty} \mathbb{E}_{\pi}\left[\frac{1}{T}\sum_{t=0}^{T}\sum_{i=1}^{N} c_i(x(t),\pi(x(t)))\right] \\ &= \sum_{i=1}^{N} \lim_{T\to\infty} \mathbb{E}_{\pi}\left[\frac{1}{T}\sum_{t=0}^{T} c_i(x(t),\pi(x(t)))\right] = \sum_{i=1}^{N} \lambda_i^{\pi}\end{aligned}$$

Continuing in this fashion, we define Q_i^{π} as the action-value function for the individual costs c_i by writing

$$\begin{aligned}Q^{\pi}(x,u) &= \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty}(c(x(t),u(t)) - \lambda^{\pi}) \mid x(0)=x, u(0)=u\right] \\ &= \sum_{i=1}^{N} \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty}(c_i(x(t),u(t)) - \lambda_i^{\pi}) \mid x(0)=x, u(0)=u\right] \\ &:= \sum_{i=1}^{N} Q_i^{\pi}(x,u)\end{aligned}$$

Doing the same for the value function lets us define V_i^{π}

$$\begin{aligned}V^{\pi}(x) &= \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty}(c(x(t),\pi(x(t))) - \lambda^{\pi}) \mid x(0)=x\right] \\ &= \sum_{i=1}^{N} \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty}(c_i(x(t),\pi(x(t))) - \lambda_i^{\pi}) \mid x(0)=x\right] \\ &:= \sum_{i=1}^{N} V_i^{\pi}(x)\end{aligned} \tag{2.12}$$

Treating the first term of the summation separately, we see that the Bellman equation introduced in Equation (2.4) and (2.5) still hold for the individual value and action-value functions

$$V_i^{\pi}(x) = c_i(x, \pi(x)) - \lambda_i^{\pi} + \mathbb{E}_{x' \sim P(x,\pi(x))}[V_i^{\pi}(x')], \tag{2.13}$$

$$Q_i^{\pi}(x,u) = c_i(x,u) - \lambda_i^{\pi} + \mathbb{E}_{x' \sim P(x,\pi(x))}[Q_i^{\pi}(x', \pi(x'))] \tag{2.14}$$

3

Problem Formulation

In this work we consider a multi-agent version of the LQR problem where the agents are embedded on a graph. Additionally, there is a spatially exponentially decaying structure between agents. We begin by specifying the dynamics, cost function and assumptions in our model. Subsequently, we introduce the spatially decaying structure and provide formal definitions for the problems under consideration.

3.1 Problem Setup

Consider an infinite-horizon, discrete time, network LQR problem controlled by N agents, $[N] := \{1, \dots, N\}$, embedded on an undirected graph. The graph is equipped with a distance function $\text{dist}(\cdot,\cdot) : [N] \times [N] \to \mathbb{R}$.

DEFINITION 3.1
A real valued function, $\text{dist}(\cdot,\cdot) : [N] \times [N] \to \mathbb{R}$ is called a distance function, if the following properties hold for all $i, j \in [N]$.

1. $\text{dist}(i,i) = 0$ and $\text{dist}(i,j) > 0$ if $i \neq j$
2. $\text{dist}(i,j) = \text{dist}(j,i)$
3. $\text{dist}(i,j) \leq \text{dist}(i,k) + \text{dist}(k,j)$ □

Since we are considering problems where the agents can be seen as being embedded on a graph, we naturally let $\text{dist}(\cdot,\cdot)$ denote the graph distance, i.e. the shortest distance between any two nodes in the graph. We will however, keep in mind that our results hold for all distance functions. The graph distance allows us to introduce the concept of the κ-neighbourhood of Agent i which we denote by $\mathcal{N}_i^{\kappa}$ and define by

$$\mathcal{N}_i^{\kappa} := \{j \in [N], \text{dist}(i,j) < \kappa\}$$

With the graph in place, we now turn to the dynamics and costs in the model. The global state x and control action u are given by

$$x(t) = [x_1(t)^\top, x_2(t)^\top, \dots, x_N(t)^\top]^\top \in \mathbb{R}^n$$
$$u(t) = [u_1(t)^\top, u_2(t)^\top, \dots, u_N(t)^\top]^\top \in \mathbb{R}^m$$

where $x_i(t) \in \mathbb{R}^{n_i}$ and $u_i(t) \in \mathbb{R}^{m_i}$ with $n = \sum_i n_i$ and $m = \sum_i m_i$ respectively. Furthermore, the state at time $t+1$ is a linear function of the previous state, control action and Gaussian system noise w.

$$x(t+1) = Ax(t) + Bu(t) + w(t), \quad w(t) \sim \mathcal{N}(0, \sigma_w^2 I)$$

As previously mentioned the global state and action spaces can be partitioned into local states and actions. For Agent i, the state at time $t+1$ is given by

$$x_i(t+1) = \sum_{j=1}^{N} [A]_{ij} x_j(t) + [B]_{ij} u_j(t)$$

where $x_i(t) \in \mathbb{R}^{n_i}$ is the local state and $u_i(t) \in \mathbb{R}^{m_i}$ is the local control action. Here $[X]_{ij}$ denotes the submatrix of X where the row indices correspond to the indices of Agent i and its column indices correspond to the indices of Agent j. By indices of Agent i, if the total index length is $n = \sum_i n_i$, we mean the indices in the range $[\sum_{j=1}^{i-1} n_j + 1, \sum_{j=1}^{i} n_j]$. Furthermore we let $[X]_{i:}$ and $[X]_{:i}$ denote the set of rows and columns corresponding to Agent i respectively.

For each agent, there is also a quadratic local cost which only depend on the state and action of the agent itself

$$c_i(t) = x_i(t)^\top [S]_{ii} x_i(t) + u_i(t)^\top [R]_{ii} u_i(t)$$

with $[S]_{ii} \in \mathbb{R}^{n_i \times n_i}$ and $[R]_{ii} \in \mathbb{R}^{m_i \times m_i}$. The global cost is defined as the summation of the individual costs

$$c(t) = \sum_{i=1}^{N} c_i(t) = x(t)^\top S x(t) + u(t)^\top R u(t)$$

Here $S \in R^{n \times n}$ and $R \in R^{m \times m}$ are both assumed to be positive semidefinite matrices. Restricting ourselves to static feedback linear policies of the form $u(t) = Kx(t)$, the problem can be formulated as a classical LQR problem

$$\begin{aligned} \min_K \quad & J(K) := \lim_{T \to \infty} \mathbb{E}\left[\frac{1}{T} \sum_{t=0}^{T-1} c(t)\right] \\ s.t. \quad & x(t+1) = Ax(t) + Bu(t) + w(t), \quad w(t) \sim \mathcal{N}(0, \sigma_w^2 I) \\ & u(t) = Kx(t) \end{aligned} \tag{P}$$

3.2 Spatially Exponentially Decaying Structure

The problem we consider here is a special type of LQR problem in which the individual agents' costs have been decoupled and where the dynamics are unknown but satisfy a spatially decaying structure introduced in [30]. The decaying property we will work with can be defined in terms of norms on the subsystems and we let $||\cdot||$ denote both the l_2-norm of a vector and the induced l_2-norm of a matrix.

DEFINITION 3.2—SPATIALLY EXPONENTIALLY DECAYING (SED)
Given a matrix $X \in \mathbb{R}^{\sum_{i=1}^{N} n_i \times \sum_{i=1}^{N} m_i}$ partitioned into $N \times N$ blocks, $[X]_{ij} \in \mathbb{R}^{n_i \times m_j}$, and distance function $\text{dist}(\cdot,\cdot) : [N] \times [N] \to \mathbb{R}$, the block matrix X is $(c,\gamma) - SED$ if

$$\left\| [X]_{ij} \right\| \leq c \cdot e^{-\gamma \text{dist}(i,j)}, \ \forall \ i,j \in [N]$$

□

Definition 3.2 aim to describe interconnected systems where the dependence between any two agents is negligible if the distance between them is large enough. In the special case of scalar blocks ($n_i = m_i = 1$), we note that $[X]_{ij}$ is a scalar and the matrix norm in Definition 3.2 reduce to the absolute value.

Before continuing, we also remark that all matrices fulfill Definition 3.2 for some γ and c. Definition 3.2 gives an upper bound on the norm of the submatrices for a given matrix and in order for this bound to be useful, c should preferably be small and γ large, relative to the matrix size.

We now provide an example to strengthen the intuition behind Definition 3.2. As previously mentioned, we let the distance be the graph distance and thus first need to define the graph topology. Consider a graph consisting of $N = 20$ chained agents, i.e., Agent 1 is connected to Agent 2, Agent 2 is connected to Agent 1 and 3 and so on (see Figure 3.1).

Figure 3.1 Graph topology for a chain of N agents.

When the agents are chained as in Figure 3.1, a SED matrix decays away from the diagonal. Schematically this is visualised in Figure 3.2. The scale is not of any importance per se, but rather Figure 3.2 visualise the rate of decay.

Using the definition, we can show how the SED parameters behaves under addition and mulitplication.

LEMMA 3.1
Suppose $X, Y \in \mathbb{R}^{n \times m}$ are $(c_x, \gamma_x) - SED$ and $(c_y, \gamma_y) - SED$, respectively, and let $\gamma = \min(\gamma_x, \gamma_y)$. Then, $X + Y$ is $(c_x + c_y, \gamma) - SED$.

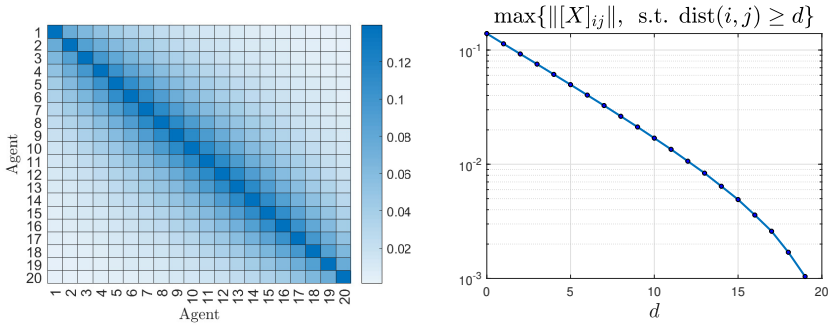
Proof.

$$\left\|[X+Y]_{ij}\right\| \leq \left\|[X]_{ij}\right\| + \left\|[Y]_{ij}\right\| \leq (c_x + c_y)e^{-\gamma \text{dist}(i,j)} \qquad \square$$

LEMMA 3.2—LEMMA 18 IN [30]
Suppose $X \in \mathbb{R}^{n\times m}$, $Y \in \mathbb{R}^{m\times p}$ are $(c_x, \gamma_x) - SED$ and $(c_y, \gamma_y) - SED$, respectively, and let $\gamma = \min(\gamma_x, \gamma_y)$. Then, XY is $(Nc_xc_y, \gamma) - SED$.

Proof.

$$\begin{aligned}
\left\|[XY]_{ij}\right\| &= \left\|\sum_{r=1}^{N}[X]_{ir}[Y]_{rj}\right\| \leq \sum_{r=1}^{N}||[X]_{ir}||||[Y]_{rj}|| \\
&\leq \sum_{r=1}^{N} c_xc_ye^{-\gamma(\text{dist}(i,r)+\text{dist}(r,j))} \leq \sum_{r=1}^{N} c_xc_ye^{-\gamma\text{dist}(i,j)} \\
&= Nc_xc_ye^{-\gamma\text{dist}(i,j)}
\end{aligned} \qquad \square$$



(a) **(b)**

Figure 3.2 Heatmap and decay plot for a SED matrix, X, when the network consists of chained agents as in Figure 3.1.

With the SED-property defined, we also define the stronger concept of spatially exponentially decaying away from Agent i.

DEFINITION 3.3
Given a matrix $X \in \mathbb{R}^{\Sigma_i n_i \times \Sigma_i m_i}$ partitioned into $N \times N$ blocks, $[X]_{ij} \in \mathbb{R}^{n_i \times m_j}$, and distance function $\text{dist}(\cdot,\cdot) : [N] \times [N] \to \mathbb{R}$, the block matrix X is (c, γ)-*SED away from i*, if $i \in [N]$ and

$$||[X]_{lj}|| \leq c \cdot e^{-\gamma \max(\text{dist}(i,l),\text{dist}(i,j))}, \ \forall l, j \in [N]$$

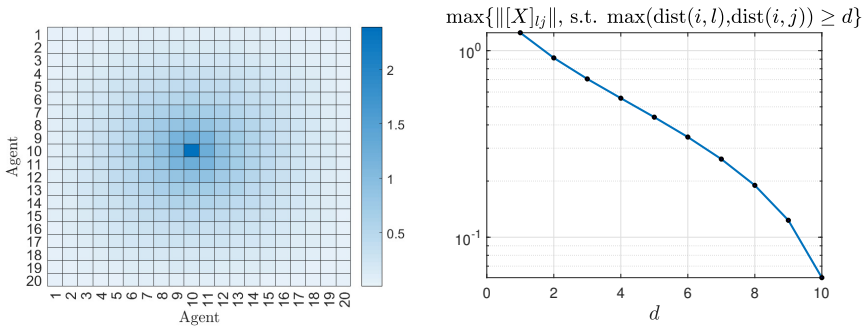
$\square$

Definition 3.3 is stronger than Definition 3.2, since

$$\max(\text{dist}(i,l),\text{dist}(i,j)) \geq \frac{\text{dist}(i,l)+\text{dist}(i,j)}{2} \geq \frac{\text{dist}(l,j)}{2}$$

meaning that if X is (c,γ)-*SED away from* i it is also $(c,\gamma/2)-SED$.

For the same chain graph topology as in Figure 3.1, an example of a matrix that is SED away from i is visualised in Figure 3.3 when $i = 10$.



(a) **(b)**

Figure 3.3 Heatmap and decay plot for a matrix, X, that is SED away from Agent 10 when the network consists of chained agents as in Figure 3.1.

By following the proofs of Lemma 3.1 and Lemma 3.2, it is easy to show that these lemmas also hold for matrices that are SED away from i. Another property of matrices that fulfill Definition 3.3 is that the decay away from i is preserved when such a matrix is multiplied by a SED matrix.

LEMMA 3.3
Let $Y \in \mathbb{R}^{n\times m}$ be $(c_y,\gamma_y)-SED$ and let $X \in \mathbb{R}^{m\times p}$ be (c_x,γ_x)-*SED away from* i. Furthermore, let $\gamma = \min(\gamma_x,\gamma_y)$. Then XY is (Nc_xc_y,γ)-*SED away from* i.

Proof.

$$\begin{aligned}
||[XY]_{lj}|| &= \left|\left|\sum_{r=1}^{N}[X]_{lr}[Y]_{rj}\right|\right| \leq \sum_{r=1}^{N}||[X]_{lr}||\,||[Y]_{rj}|| \\
&\leq \sum_{r=1}^{N} c_xc_ye^{-\gamma(\max(\text{dist}(i,l),\text{dist}(i,r))+\text{dist}(r,j))} \\
&\leq Nc_xc_ye^{-\gamma\max(\text{dist}(i,l),\text{dist}(i,j))}
\end{aligned}$$

Where we used that $\max(a+c,b+c) \geq \max(a,b+c)\ \forall\ a,b,c \geq 0$ and the triangle inequality in the last step. □

We notice that there is nothing special about multiplying Y by X from the left and it is easy to see that the same bound holds if we change the dimensions of X or Y and consider the product YX.

We now return to problem (P) and note that, since the costs are decoupled, the S and R matrices are sparse and trivially SED. The matrices A and B are however dense in general and we add the following decaying assumption on the dynamics

ASSUMPTION 1—SED DYNAMICS
There exist $\gamma_{sys} > 0$ and constants $c_A, c_B > 0$ such that A, B are $(c_A, \gamma_{sys}), (c_B, \gamma_{sys})$-SED repsectively. Without loss of generality we assume $c_A, c_B \geq 1$.

Under Assumption 1, we constrain ourselves to systems where the interdependence in the dynamics between agents becomes exponentially small as the distance between them increases.

As previously mentioned, if we are allowed to pick γ and c freely, Assumption 1 will hold for any finite-dimensional matrix. In order to get a useful bound from Assumption 1, it is therefore important that c and γ do not scale with N. Meaning, there exists some upper and lower bounds on c and γ that are independent of N. This property can be relaxed to include the case when there is a dependence on N, as long as c grows, or γ decays slowly as N increases.

3.3 Stable Systems and Stabilising Controllers

Another useful concept we will need is the concept of stability and especially stabilising policies. Intuitively, a system is stable if and only if any bounded input produces a bounded output. Formally, we define the concept of (τ, ρ)-stability by

DEFINITION 3.4—(τ, ρ)-STABILITY
For $\tau \geq 1,\ \rho > 0$, a matrix X is said to be (τ, ρ)-stable if $||X^k|| \leq \tau \cdot e^{-\rho k},\ \forall\, k \in \mathbb{Z}_{\geq 0}$

By [10, Theorem 5.6.12 and Corollary 5.6.13] the existence of such a τ and ρ for the matrix X is equivalent to the standard definition of stability, which states that the spectral radius of X is less than 1. Definition 3.4 is however stronger than the standard definition, since it explicitly provides the rate of convergence. We say that a policy K is stabilising if there exists $\tau,\ \rho$ such that the closed-loop system $(A + BK)$ is (τ, ρ)-stable.

3.4 Structure of the Optimal Controller

Previously the structure of the optimal controller for network LQR with spatial exponential decay has been studied by Zhang, Li and Li in [30]. That work considers

problem (P) under Assumption 1 and the additional assumption that there exists an initial stabilising controller, K_0 which is $(k_0, \gamma_{sys}) - SED$. Under these assumptions they show that the optimal controller also has a decaying structure, where the norm of the subsystems of the optimal controller $||[K^*]_{ij}||$, are $O\left(\exp\left(\frac{-c\cdot \text{dist}(i,j)}{\text{poly}\ln(N)}\right)\right)$ [30]. They continue studying truncated controllers $\underline{K}^\kappa$ of the form

$$[\underline{K}^\kappa]_{ij} = \begin{cases} [K^*]_{ij} & \text{if } \text{dist}(i,j) < \kappa \\ 0 & \text{otherwise} \end{cases}$$

and show that by taking $\kappa \sim \text{poly}\ln(N)\ln(1/\varepsilon)$, $\underline{K}^\kappa$ achieves ε-optimal control [30]. This means that the optimal controller can be well approximated by a truncated version, allowing for distributed control of the system.

3.5 Problem Statement

We are mainly concerned with the problem of using reinforcement learning to learn a sample based controller for network LQR with SED structure. Motivated by the previous results described in Section 3.4, we aim to address two questions: Firstly, how the decaying structure is preserved for the individual value and action-value functions; Secondly, can we use the structure of the problem to design a distributed reinforcement learning algorithm to learn distributed controllers. More formally, we want to address the following two problems:

Problem 1. Consider stabilising, spatially exponentially decaying, linear feedback policies K, that is, K such that, K is $(c_K, \gamma_K) - SED$ and $(A + BK)$ is (τ, ρ)-stable for some c_K, γ_K, τ and ρ. Does the individual value function V_i and action-value function Q_i for problem (P) also have a spatially decaying structure under decoupled costs (Equation (2.11)) and Assumption 1?

Problem 2. Let $\mathscr{K}^\kappa$ be a class of localised controllers defined by

$$\mathscr{K}^\kappa := \{K \in \mathbb{R}^{m\times n} : [K]_{ij} = 0_{m_i \times n_j} \text{ if } j \notin \mathscr{N}_i^\kappa\}.$$

In the case when A and B in (P) are unknown but satisfy Assumption 1, can the network structure be used to design a scalable RL algorithm for learning a stabilising, localised, feedback controller $K \in \mathscr{K}^\kappa$?

4

Individual Value and Action-Value Functions for Network LQR

This section aims to address Problem 1 (see page 25). Inspired by the results on the structure of the optimal controller in Section 3.4, we derive the individual value and action-value functions for problem (P) in order to investigate their structure.

4.1 Derivation of the Individual Value and Action-Value Functions in the Network LQR setting

We start by investigating the value and action-value functions for the individual agents. For a linear policy K we first note that Agent i's action at time t is given by $u_i(t) = [K]_{i:}x(t)$. We then define $S_i' \in \mathbb{R}^{n\times n}$ and $R_i' \in \mathbb{R}^{m\times m}$ to be zero-padded versions of $[S]_{ii}$ and $[R]_{ii}$, such that, $x^\top S_i' x = x_i^\top [S]_{ii} x_i$ and $u^\top R_i' u = u_i^\top [R]_{ii} u_i$. Using the definition of the individual value function from Equation (2.12) together with decoupled quadratic costs and linear dynamics we get

$$
\begin{aligned}
V_i(x) &= \mathbb{E}\left[\sum_{t=0}^{\infty}(x_i^\top(t)[S]_{ii}x_i(t) + u_i^\top(t)[R]_{ii}u_i(t) - \lambda_i) \mid x(0) = x\right] \\
&= \mathbb{E}\left[\sum_{t=0}^{\infty}(x_i^\top(t)[S]_{ii}x_i(t) + x^\top(t)[K]_{i:}^\top[R]_{ii}[K]_{i:}x(t) - \lambda_i) \mid x(0) = x\right] \\
&= \mathbb{E}\left[\sum_{t=0}^{\infty}(x^\top(t)(S_i' + [K]_{i:}^\top[R]_{ii}[K]_{i:})x(t) - \lambda_i) \mid x(0) = x\right] \\
&= x^\top(S_i' + [K]_{i:}^\top[R]_{ii}[K]_{i:})x - \lambda_i + \mathbb{E}_{\substack{x'=Ax+Bu+w \\ w\sim\mathcal{N}(0,\sigma_w^2 I)}}\left[V_i(x')\right]
\end{aligned}
$$

We notice that the individual value-functions are in fact on the same form as the global value-function given by Equation (2.6). Once again using a quadratic ansatz, it can be seen that the individual value function is also quadratic, with $V_i(x) = x^\top P_i x$ where P_i satisfies the Lyapunov equation

$$P_i = S_i' + [K]_{i:}^\top [R]_{ii} [K]_{i:} + (A+BK)^\top P_i (A+BK) \tag{4.1}$$

Similarly, for the individual action-value functions

$$\begin{aligned} Q_i(x,u) &= x^\top S_i' x + u^\top R_i' u + (Ax+Bu)^\top P_i (Ax+Bu) && (4.2)\\ &= \begin{pmatrix} x^\top & u^\top \end{pmatrix} \begin{pmatrix} S_i' + A^\top P_i A & A^\top P_i B \\ B^\top P_i A & R_i' + B^\top P_i B \end{pmatrix} \begin{pmatrix} x \\ u \end{pmatrix} && (4.3)\\ &= \begin{pmatrix} x^\top & u^\top \end{pmatrix} \begin{pmatrix} H_{i11} & H_{i12} \\ H_{i12}^\top & H_{i22} \end{pmatrix} \begin{pmatrix} x \\ u \end{pmatrix} && (4.4)\\ &= \begin{pmatrix} x^\top & u^\top \end{pmatrix} H_i \begin{pmatrix} x \\ u \end{pmatrix} && (4.5) \end{aligned}$$

The importance of the individual value and action-value functions also being quadratic is not immediate. However, this structure imply they have the same form as the global value and action-value functions where a lot more work has been done. This allows us to borrow both proof-techniques and design ideas from centralised LQR.

4.2 Spatially Decaying Structure of the Individual Value and Action-Value Functions

We now investigate the structure of V_i and Q_i from the previous section. We notice that P_i solving the Lyapunov equation (4.1) plays an important role for both the individual value and action-value functions. Hence, it is logical to commence our investigation by examining how P_i preserves spatial decay. We start by considering a more general Lyapunov equation.

LEMMA 4.1
Let $L \in \mathbb{R}^{n\times n}$ be (τ,ρ)-stable and $(c_L,\gamma)-SED$, with $c_L \geq 1$, and $M \in \mathbb{R}^{n\times n}$ (c_M,γ)-*SED away from i*, then the solution P to the Lyapunov equation $P = L^\top P L + M$, is (c_P,γ_P)-*SED away from i* with

$$c_P = \frac{||M||\tau^2}{1-e^{-2\rho}} + 2c_M$$

and

$$\gamma_P = \frac{\rho\gamma}{\rho + \ln(Nc_L)}$$

□

Proof. Since L is stable, the solution to the Lyapunov equation is unique and given by

$$P = \sum_{k=0}^{\infty} (L^k)^\top M L^k$$

Define P^t to be the first t terms in the series

$$P^t = \sum_{k=0}^{t-1} (L^k)^\top M L^k$$

then using that L is (τ, ρ)-stable,

$$\begin{aligned}
\left\|P - P^t\right\| &= \left\| \sum_{k=t}^{\infty} (L^\top)^k M L^k \right\| \leq \sum_{k=t}^{\infty} ||(L^\top)^k||\ ||M||\ ||L^k|| \\
&\leq ||M|| \sum_{k=t}^{\infty} \tau^2 e^{-2\rho k} = ||M|| \sum_{k=0}^{\infty} \tau^2 e^{-2\rho(k+t)} \\
&= \frac{||M||\tau^2}{1 - e^{-2\rho}} e^{-2\rho t}
\end{aligned}$$

Now since L is $(c_L, \gamma) - SED$, Lemma 3.2 says L^k is $(c_{L^k}, \gamma) - SED$, where $c_{L^k} = N^{k-1} c_L^k$ and by Lemma 3.3 it holds that

$$||[ML^k]_{lj}|| \leq (Nc_L)^k c_M e^{-\gamma \max(\text{dist}(i,l), \text{dist}(i,j))}$$

and

$$||[(L^k)^\top M L^k]_{lj}|| \leq (Nc_L)^{2k} c_M e^{-\gamma \max(\text{dist}(i,l), \text{dist}(i,j))}$$

Furthermore, $c_L \geq 1$ implies $(Nc_L)^2 \geq 2$ and

$$\sum_{k=0}^{t-1} (Nc_L)^{2k} c_M = c_M \frac{(Nc_L)^{2t} - 1}{(Nc_L)^2 - 1} \leq 2(Nc_L)^{2(t-1)}$$

this means we can bound $||[P^t]_{lj}||$ by

$$||[P^t]_{lj}|| \leq 2c_M (Nc_L)^{2(t-1)} e^{-\gamma \max(\text{dist}(i,l), \text{dist}(i,j))}$$

Combining these results gives

$$\begin{aligned}||[P]_{lj}|| &\leq ||[P-P^t]_{lj}|| + ||[P^t]_{lj}|| \\ &\leq ||P-P^t|| + 2c_M(Nc_L)^{2(t-1)}e^{-\gamma\max(\text{dist}(i,l),\text{dist}(i,j))} \\ &\leq \frac{||M||\tau^2}{1-e^{-2\rho}}e^{-2\rho t} + 2c_M(Nc_L)^{2(t-1)}e^{-\gamma\max(\text{dist}(i,l),\text{dist}(i,j))}\end{aligned}$$

This holds for any t, in particular it holds when the two terms are roughly equal. That is, for t such that

$$e^{-2\rho t} = (Nc_L)^{2(t-1)}e^{-\gamma\max(\text{dist}(i,l),\text{dist}(i,j))}$$

and we therefore set

$$t = \left\lfloor \frac{\gamma\max(\text{dist}(i,l),\text{dist}(i,j))}{2(\rho+\ln(Nc_L))} \right\rfloor + 1$$

which gives

$$\begin{aligned}||[P]_{lj}|| &\leq \frac{||M||\tau^2}{1-e^{-2\rho}}e^{-2\rho t} + 2c_M(Nc_L)^{2(t-1)}e^{-\gamma\max(\text{dist}(i,l),\text{dist}(i,j))} \\ &\leq \left(\frac{||M||\tau^2}{1-e^{-2\rho}} + 2c_M\right)\exp\left(-\frac{\rho\gamma}{\rho+\ln(Nc_L)}\max(\text{dist}(i,l),\text{dist}(i,j))\right) \quad \square\end{aligned}$$

Lemma 4.1 states that the Lyapunov equation preserves the spatial decay away from i when the matrix L is (τ,ρ)-stable. The proof is essentially borrowed from [30] where they prove a similar result with M being SED, here adapted to the case when M is SED away from i. We remark that the condition $c_L \geq 1$ is not strictly necessary but allows for a tidier bound and was added for convenience in the proof. In order for Lemma 4.1 to provide a useful bound, it is important that the parameters $\gamma,\rho,\tau,||M||$, etc, do not scale with N. Due to the $\ln(Nc_L)$ term, the exponent always scale with N, which worsens the bound as N grows. However, the rate of growth is slow and as long as $\max(\text{dist}(i,l),\text{dist}(i,j)) \geq N^\varepsilon$ for any $\varepsilon > 0$, $[P]_{lj} \to 0$ as $N \to \infty$.

Prior to stating our main result, we state another lemma about the structure of $S_i' + [K]_{i:}^\top [R]_{ii}[K]_{i:}$ showing up in Equation (4.1).

LEMMA 4.2
Let K be $(c_K,\gamma_{sys})-SED$. Then, the matrix $S_i' + [K]_{i:}^\top [R]_{ii}[K]_{i:}$ from the Lyapunov equation (4.1) is $(|||[S]_{ii}|| + |||[R]_{ii}||c_K^2,\gamma_{sys})$-*SED away from* i.

Proof. First we note that

$$\left\| [[K]_{i:}^\top [R]_{ii}[K]_{i:}]_{lj} \right\| \leq |||[R]_{ii}||c_K^2 e^{-\gamma_{sys}\text{dist}(i,l)}e^{-\gamma_{sys}\text{dist}(i,j)}$$

and since $e^{-\gamma_{sys}\text{dist}(i,l)} \leq 1$ for all l we get

$$\begin{aligned}\left|\left|[[K]_{i:}^\top [R]_{ii}[K]_{i:}]_{lj}\right|\right| &\leq ||[R]_{ii}||c_K^2 \min(e^{-\gamma_{sys}\text{dist}(i,l)}, e^{-\gamma_{sys}\text{dist}(i,j)}) \\ &= ||[R]_{ii}||c_K^2 e^{-\gamma_{sys}\max(\text{dist}(i,l),\text{dist}(i,j))}\end{aligned}$$

By construction of S_i' we know that $[S_i']_{lj} = 0$ unless $j = l = i$ and thus

$$||[S_i']_{lj}|| \leq ||[S]_{ii}||e^{-\gamma\max(\text{dist}(i,l),\text{dist}(i,j))}$$

is true for any γ, in particular it is true for $\gamma = \gamma_{sys}$. Combining these results using Lemma 3.1 into

$$||[S_i' + [K]_{i:}^\top [R]_{ii}[K]_{i:}]_{lj}|| \leq (||[S]_{ii}|| + ||[R]_{ii}||c_K^2)e^{-\gamma_{sys}\max(\text{dist}(i,l),\text{dist}(i,j))}$$

finishes the proof. □

We are now ready for our main result concerning the structure of P_i that solves Equation (4.1)

THEOREM 4.3
Let K be $(c_K, \gamma_{sys}) - SED$ and such that the closed system $(A+BK)$ is (τ,ρ)-stable. Then for the solution P_i to the Lyapunov equation

$$P_i = S_i' + [K]_{i:}^\top [R]_{ii}[K]_{i:} + (A+BK)^\top P_i (A+BK)$$

it holds that P_i is (c_{P_i}, γ_{P_i})-*SED away from i*, with

$$c_{P_i} = \frac{||S_i' + [K]_{i:}^\top [R]_{ii}[K]_{i:}||\tau^2}{1-e^{-2\rho}} + 2(||[S]_{ii}|| + ||[R]_{ii}||c_K^2)$$

and

$$\gamma_{P_i} = \frac{\rho\gamma_{sys}}{\rho + \ln(Nc_A + N^2 c_B c_K)}.$$

□

Proof. Lemma 3.2 and 3.1 gives that $(A+BK)$ is $(c_A + Nc_B c_K, \gamma_{sys}) - SED$ and Lemma 4.2 tells us the decay rate of $S_i' + [K]_{i:}^\top [R]_{ii}[K]_{i:}$. The result directly follows by setting $L = (A+BK)$ and $M = S_i' + [K]_{i:}^\top [R]_{ii}[K]_{i:}$ in Lemma 4.1. □

Theorem 4.3 says that the individual value-functions $V_i(x) = x^\top P_i x$ dependency on other agents decays exponentially as the distance between them increases. We remark that the result in Theorem 4.3 is rather conservative and the bound is not tight.

By revisiting the definition of the submatrices H_{i11}, H_{i12} and H_{i22} parameterising Q_i and using Lemma 3.3, we get a similar result for the individual action-value function.

COROLLARY 4.4
For a linear policy fulfilling the assumptions in Theorem 4.3, the submatrices H_{i11}, H_{i12} and H_{i22} of the matrix H_i defined in Equation (4.4) are all (c_{H_i}, γ_{P_i})-*SED away from i*, with

$$c_{H_i} = \max\left(c_{S_i'} + N^2 c_A^2 c_{P_i},\ N^2 c_A c_B c_{P_i},\ c_{R_i'} + N^2 c_B^2 c_{P_i}\right) \qquad \square$$

Proof. The result follows immediately by applying Lemma 3.1 and 3.2 on the definition of the submatrices and then choosing c_{H_i} as the maximum coefficient. $\square$

From Equation (4.4), we know that the individual action-value functions depend on the global states and actions. In order to design a scalable algorithm that utilises the individual action-value functions, we want to remove the global dependence. It is thus interesting to bound the error caused by truncating these functions. We first define the κ-truncated matrices.

DEFINITION 4.1

$$[\underline{H}_{i11}^{\kappa}]_{lj} := \begin{cases} [H_{i11}]_{lj} & \text{if } \max(\text{dist}(i,l), \text{dist}(i,j)) < \kappa \\ 0 & \text{otherwise} \end{cases}$$

$$[\underline{H}_{i12}^{\kappa}]_{lj} := \begin{cases} [H_{i12}]_{lj} & \text{if } \max(\text{dist}(i,l), \text{dist}(i,j)) < \kappa \\ 0 & \text{otherwise} \end{cases}$$

$$[\underline{H}_{i22}^{\kappa}]_{lj} := \begin{cases} [H_{i22}]_{lj} & \text{if } \max(\text{dist}(i,l), \text{dist}(i,j)) < \kappa \\ 0 & \text{otherwise} \end{cases}$$

$\square$

Using Corollary 4.4 we can now bound the error caused by truncating H_{i11}, H_{i12} and H_{i22} with the following lemma.

LEMMA 4.5
For a linear policy fulfilling the assumptions in Theorem 4.3 the error caused by truncating the submatrices of H_i is bounded by

$$||H_{i11} - \underline{H}_{i11}^{\kappa}||, ||H_{i12} - \underline{H}_{i12}^{\kappa}||, ||H_{i22} - \underline{H}_{i22}^{\kappa}|| \leq \sqrt{N} c_{H_i} e^{-\gamma_{P_i}\kappa}$$

$\square$

Proof. From the definition of $\underline{H}_{i11}^{\kappa}$ and Corollary 4.4 we have that

$$||[H_{i11} - \underline{H}_{i11}^{\kappa}]_{lj}|| \leq c_{H_i} e^{-\gamma_{P_i}\kappa}$$

Multiplying by $x \in \mathbb{R}^n$ with $||x|| = 1$ gives

$$||[(H_{i11} - \underline{H}^{\kappa}_{i11})x]_l|| = \left|\left|\sum_{r=1}^{N}[(H_{i11} - \underline{H}^{\kappa}_{i11})]_{lr}x_r\right|\right| \leq c_{H_i}e^{-\gamma_{P_i}\kappa}\sum_{r=1}^{N}||x_r||$$

and thus

$$\begin{aligned}||(H_{i11} - \underline{H}^{\kappa}_{i11})x||^2 &\leq N(c_{H_i}e^{-\gamma_{P_i}\kappa})^2\left(\sum_{r=1}^{N}||x_r||\right)^2 \\ &\leq N(c_{H_i}e^{-\gamma_{P_i}\kappa})^2\sum_{r=1}^{N}||x_r||^2 \\ &= N(c_{H_i}e^{-\gamma_{P_i}\kappa})^2||x||^2\end{aligned}$$

Finally taking square roots on each side gives the result for H_{i11}. Repeating the same procedure for H_{i12} and H_{i22} finishes the proof. □

Lemma 4.5 implies that in order to truncate the submatrices of H_i with an $\varepsilon > 0$ error, taking $\kappa \geq \ln\left(\frac{\sqrt{N}c_{H_i}}{\varepsilon}\right)/\gamma_{P_i}$ is sufficient. Once again, we emphasise the importance of checking the dependence of the different parameters on N. As an extreme example, consider a situation where c_{H_i} grows exponentially with N and $\gamma_{P_i} \leq 1$. Then, $\kappa \geq N$ is required to keep the error upper bounded by ε, meaning we are not able to truncate the action-value function. This is, however, not generally the case of interest, and we can instead use Lemma 4.5 to give a reasonably small bound on the error due to truncation.

5

Scalable MARL for Network LQR

Inspired by the findings from the previous chapter, we continue to design an algorithm where learning the truncated individual action-value functions serves as an intermediate step. Building on the actor-critic framework, we design an algorithm where the critic estimates the individual action-value functions. Based on these estimates, the actor calculates the gradient of the cost function and updates the parameters by taking a step towards a lower cost. The actor and the critic are two separate architectures, and we provide a detailed description of each architecture before combining them. First, however, we give some background on a method for centralised learning of LQR.

5.1 Least-Squares Temporal Difference Learning for Centralised Q-Function Evaluation

Linear-quadratic control has been studied extensively from the reinforcement learning perspective, particularly as a test-bed for learning algorithms in continuous control [1, 5, 7, 14, 21, 26]. The reason is that both the action space and state space are continuous and that the optimal controller is known, enabling easy quantification of learning algorithms' performance [21]. Moreover, the problem is theoretically tractable due to its simplicity, allowing guarantees to be provided for these algorithms [26]. Importantly, many of the observed patterns persist when considering more challenging nonlinear problems [21].

Important for our work is that in [14], which applies the method developed in [15] to learn the action-value function for the linear-quadratic control problem and then take the next action greedily. After deriving their method, their analysis proves that approximately $(n+m)^3\varepsilon^{-2}\log(1/\varepsilon)$ samples are required to learn an ε-optimal static feedback controller. The critic in our method can be described as a distributed version of the LSTDQ estimator for estimating individual action-value

functions rather than the global action-value function. Here, we introduce the theory behind their algorithm and refer the interested reader to [14].

In order to do this, we first introduce some notation. For a symmetric matrix $M \in \mathbb{R}^{n\times n}$, $\text{svec}(M) \in \mathbb{R}^{n(n+1)/2}$ denotes the vectorised version of the upper triangular part of M so that $\text{svec}(M)^\top \text{svec}(M) = ||M||_F^2$. We let $\text{smat}(\cdot)$ be the inverse of $\text{svec}(\cdot)$ so that $\text{smat}(\text{svec}(M)) = M$. Using this we recall Equation (2.7) and write

$$Q(x,u) = \begin{pmatrix} x^\top & u^\top \end{pmatrix} H \begin{pmatrix} x \\ u \end{pmatrix} = \text{svec}(H)^\top \text{svec}\left(\begin{pmatrix} x \\ u \end{pmatrix} \begin{pmatrix} x \\ u \end{pmatrix}^\top \right) := h^\top \phi(x,u) \quad (5.1)$$

The linearity of the action-value function allows the use of the Least-squares temporal difference learning (LSTDQ) method developed in [15]. LSTDQ approximates the action-value function by forcing the approximated function to be a fixed point to the Bellman equation (2.5). This is achieved by projecting the estimate onto the subspace spanned by the features $\phi(x,u)$.

Formally, the authors of [14] assume they have access to a sample trajectory $\{(x(t),u(t),Kx(t+1))\}_{t=1}^T$ where $\{u(t)\}$ can be any arbitrary sequence, as long as it ensures sufficient exploration [14]. They then introduce,

$$\phi(t) := \phi(x(t),u(t)), \quad (5.2)$$
$$\psi(t) := \phi(x(t),Kx(t)), \quad (5.3)$$
$$c(t) := x(t)^\top S x(t) + u(t)^\top R u(t), \quad (5.4)$$
$$f := \text{svec}\left(\sigma_w^2 \begin{pmatrix} I \\ K \end{pmatrix} \begin{pmatrix} I \\ K \end{pmatrix}^\top \right) \quad (5.5)$$

and estimate the parameters of the action-value function with

$$\widehat{h} := \left(\sum_{t=1}^T \phi(t) \left(\phi(t) - \psi(t+1) + f \right)^\top \right)^\dagger \sum_{t=1}^T \phi(t) c(t) \quad (5.6)$$

where $M^\dagger$ denotes the pseudo-inverse of a real valued matrix M. After estimating the parameters of the action-value function with Equation (5.6), their algorithm updates the policy greedily by minimising the approximated action-value function.

5.2 Critic Architecture: Decentralised Q-function estimation

Inspired by the result of Lemma 4.5, we adapt the LSTDQ algorithm from [14] to learn truncated Q_i in a decentralised way. We consider stabilising, decentralised

policies $K \in \mathscr{K}^{\kappa}$ (see page 25) and let $x_{\mathscr{N}_i^{\kappa}}$ denote the neighbourhood state of Agent i meaning the concatenation of all x_j with $j \in \mathscr{N}_i^{\kappa}$ and similar for $u_{\mathscr{N}_i^{\kappa}}$. We let $\underline{\cdot}$ denote truncation and $\widehat{\cdot}$ denote estimation from samples. Using Definition 4.1, we introduce the truncated individual action-value function by

$$\underline{Q}_i(x,u) := \begin{pmatrix} x^\top & u^\top \end{pmatrix} \begin{pmatrix} \underline{H}^{\kappa}_{i11} & \underline{H}^{\kappa}_{i12} \\ \underline{H}^{\kappa\top}_{i12} & \underline{H}^{\kappa}_{i22} \end{pmatrix} \begin{pmatrix} x \\ u \end{pmatrix} = \underline{Q}_i(x_{\mathscr{N}_i^{\kappa}}, u_{\mathscr{N}_i^{\kappa}})$$

Lemma 4.5 then suggests that for sufficiently large κ

$$Q_i(x,u) \approx \underline{Q}_i(x,u) = \underline{Q}_i(x_{\mathscr{N}_i^{\kappa}}, u_{\mathscr{N}_i^{\kappa}})$$

Using the feature vector $\phi(x,u)$ introduced in Equation (5.1) we write

$$\underline{Q}_i(x_{\mathscr{N}_i^{\kappa}}, u_{\mathscr{N}_i^{\kappa}}) = h_i^\top \phi(x_{\mathscr{N}_i^{\kappa}}, u_{\mathscr{N}_i^{\kappa}})$$

Our later proposed algorithm will be an off-policy algorithm which means the policy used to generate data is different from the one being learned [25]. Thus the input u^0 can come from any sequence, as long as it provides sufficient exploration. We will consider inputs of the form

$$u^0(t) = K_0 x(t) + \eta(t),\ \eta(t) \sim \mathscr{N}(0, \sigma_\eta^2 I)$$

where $K_0 \in \mathscr{K}^{\kappa}$ is a stabilising initial policy and η injected noise in order to ensure sufficient exploration. In order to evaluate the current policy we also need actions from our current policy $K \in \mathscr{K}^{\kappa}$

$$u^K(t) = Kx(t)$$

Since both K_0 and K belong to $\mathscr{K}^{\kappa}$ they are localised policies in the sense that Agent i's next action only depend on the states of agents in $\mathscr{N}_i^{\kappa}$. Now suppose Agent i has access to a trajectory, $D_{\mathscr{N}_i^{\kappa}}$ consisting of T_c samples

$$D_{\mathscr{N}_i^{\kappa}} := \left\{ \left(x_{\mathscr{N}_i^{\kappa}}(t), u^0_{\mathscr{N}_i^{\kappa}}(t), u^K_{\mathscr{N}_i^{\kappa}}(t+1) \right) \right\}_{t=1}^{T_c}$$

Then, similar to equations (5.2)-(5.5), we define

$$\phi_i(t) := \phi(x_{\mathscr{N}_i^{\kappa}}(t), u^0_{\mathscr{N}_i^{\kappa}}(t)), \tag{5.7}$$

$$\psi_i(t) := \phi(x_{\mathscr{N}_i^{\kappa}}(t), u^K_{\mathscr{N}_i^{\kappa}}(t)), \tag{5.8}$$

$$c_i(t) := x_i(t)^\top [S]_{ii} x_i(t) + u_i^0(t)^\top [R]_{ii} u_i^0(t), \tag{5.9}$$

$$f_i := \operatorname{svec}\left(\sigma_w^2 \begin{pmatrix} I \\ [K]_{\mathscr{N}_i^{\kappa}:} \end{pmatrix} \begin{pmatrix} I \\ [K]_{\mathscr{N}_i^{\kappa}:} \end{pmatrix}^\top \right) \tag{5.10}$$

and define the LSTDQ estimator for h_i by

$$\widehat{h}_i := \left(\sum_{t=1}^{T_c} \phi_i(t) \left(\phi_i(t) - \psi_i(t+1) + f_i \right)^\top \right)^\dagger \sum_{t=1}^{T_c} \phi_i(t) c_i(t) \tag{5.11}$$

Using $\mathrm{smat}(\cdot)$ we are able to transform the estimate back into matrix form. When doing so, we exploit that H_i is a symmetric positive semidefinite matrix [5] and project the estimate onto the set of symmetric positive semidefinite matrices. This is a closed convex set and therefore this projection can only reduce the error [1]. I.e., we define

$$\mathrm{Proj}(\cdot) := \underset{X=X^\top, X \succeq 0}{\arg\min} \; ||X - \cdot||_F \tag{5.12}$$

and form $\widehat{H}_i = \mathrm{Proj}(\mathrm{smat}(\widehat{h}_i))$. For notational convenience we form $\underline{\widehat{H}}_i$ by appending $\widehat{H}_i$ with zeros and write

$$\underline{\widehat{Q}}_i(x_{\mathcal{N}_i^\kappa}, u_{\mathcal{N}_i^\kappa}) = \begin{pmatrix} x^\top & u^\top \end{pmatrix} \underline{\widehat{H}}_i \begin{pmatrix} x \\ u \end{pmatrix} = \begin{pmatrix} x^\top & u^\top \end{pmatrix} \begin{pmatrix} \underline{\widehat{H}}_{i11} & \underline{\widehat{H}}_{i12} \\ \underline{\widehat{H}}_{i12}^\top & \underline{\widehat{H}}_{i22} \end{pmatrix} \begin{pmatrix} x \\ u \end{pmatrix}$$

where $\underline{\widehat{H}}_i$ by construction is sparse and such that

$$[\underline{\widehat{H}}_{i11}]_{lj} = 0_{n_l \times n_j}, \; [\underline{\widehat{H}}_{i12}]_{lj} = 0_{n_l \times m_j}, \; [\underline{\widehat{H}}_{i22}]_{lj} = 0_{m_l \times m_j} \text{ if } i \notin \mathcal{N}_l^\kappa \cap \mathcal{N}_j^\kappa$$

This implies,

$$[\underline{\widehat{H}}_{11}]_{ij} := \sum_{l=1}^{N} [\underline{\widehat{H}}_{l11}]_{ij} = \sum_{l \in \mathcal{N}_i^\kappa \cap \mathcal{N}_j^\kappa} [\underline{\widehat{H}}_{l11}]_{ij} \tag{5.13}$$

and similar for $\underline{\widehat{H}}_{12}$ and $\underline{\widehat{H}}_{22}$. Meaning, any parameters concerning Agent i in the estimate of the global Q-function are completely determined by the local action-value functions in the κ-neighbourhood of i. Furthermore, the global estimate is sparse since

$$[\underline{\widehat{H}}_{11}]_{ij} = 0_{n_i \times n_j}, \; [\underline{\widehat{H}}_{12}]_{ij} = 0_{m_i \times n_j}, \; [\underline{\widehat{H}}_{22}]_{ij} = 0_{m_i \times m_j} \text{ if } j \notin \mathcal{N}_i^{2\kappa-1} \tag{5.14}$$

which we see by noting that $\mathcal{N}_i^\kappa \cap \mathcal{N}_j^\kappa = \emptyset$ when $j \notin \mathcal{N}_i^{2\kappa-1}$. This is a general property stemming from the definition of $\mathcal{N}_i^\kappa$ which the following lemma proves.

LEMMA 5.1

$$\mathcal{N}_i^{2\kappa-1} = \bigcup_{j \in \mathcal{N}_i^\kappa} \mathcal{N}_j^\kappa \qquad \square$$

Proof. We use a double inclusion argument to prove the lemma.

$\mathscr{N}_i^{2\kappa-1} \supseteq \bigcup_{j\in\mathscr{N}_i^\kappa} \mathscr{N}_j^\kappa$: Assume $l \in \bigcup_{j\in\mathscr{N}_i^\kappa} \mathscr{N}_j^\kappa$, specifically let $l \in \mathscr{N}_{j_1}^\kappa$ with $j_1 \in \mathscr{N}_i^\kappa$ then $l \in \mathscr{N}_i^{2\kappa-1}$ since

$$\operatorname{dist}(l,i) \leq \operatorname{dist}(l,j_1) + \operatorname{dist}(j_1,i) \leq \kappa - 1 + \kappa - 1 = (2\kappa-1) - 1$$

$\mathscr{N}_i^{2\kappa-1} \subseteq \bigcup_{j\in\mathscr{N}_i^\kappa} \mathscr{N}_j^\kappa$: Now assume $l \in \mathscr{N}_i^{2\kappa-1}$. If $\operatorname{dist}(l,i) \leq \kappa - 1$ we are done since $l \in \mathscr{N}_i^\kappa$ by definition. Instead assume $\operatorname{dist}(l,i) > \kappa - 1$. Consider the last node $j \in \mathscr{N}_i^\kappa$ on the shortest path from l to i, that is $j \in \mathscr{N}_i^\kappa$ with $\operatorname{dist}(i,j) = \kappa - 1$. Then $l \in \mathscr{N}_j^\kappa \subseteq \bigcup_{j\in\mathscr{N}_i^\kappa} \mathscr{N}_j^\kappa$ since

$$\operatorname{dist}(j,l) = \operatorname{dist}(i,l) - \operatorname{dist}(i,j) \leq 2\kappa - 2 - (\kappa - 1) = \kappa - 1$$

holds on the shortest path. □

We use the introduced notation and formulate the critic architecture in Algorithm 1 and 2 for Agent i. The critic is the first half of our learning algorithm and its purpose is to estimate the action-value functions. In Algorithm 1, the data collection procedure is described and in Algorithm 2 the Q-function estimation procedure.

Algorithm 1 Critic: COLLECTCRITICDATA

Require:

$K_{play} \in \mathscr{K}^\kappa$: Stabilising exploration controller,

T_c: Trajectory length

σ_η^2: Exploration variance,

κ: Neighbourhood size

procedure COLLECTCRITICDATA(K_{play}, T_c, σ_η^2, κ)
  for $t = 1, \ldots, T_c + 1$ **do**
    Share $x_i(t)$ with the κ-neighbourhood.
    Calculate

$$u_i^{\text{play}}(t) = \sum_{j\in\mathscr{N}_i^\kappa} [K_{play}]_{ij} x_j(t) + \eta(t), \eta(t) \sim \mathscr{N}(0, \sigma_\eta^2 I)$$

    Share $u_i^{\text{play}}(t)$ with the κ-neighbourhood.
  end for
  Agent i now has access to

$$D_{\mathscr{N}_i^\kappa} := \bigcup_{j\in\mathscr{N}_i^\kappa} \left\{ \left(x_j(t), u_j^{play}(t) \right) \right\}_{t=1}^{T_c+1}$$

  return $D_{\mathscr{N}_i^\kappa}$ to Agent i.
end procedure

Algorithm 2 Critic: ESTIMATEQ

Require:

$D_{\mathcal{N}_i^\kappa}$: Data trajectory for all i

$K_k \in \mathscr{K}^\kappa$: Current controller

T_c: Trajectory length

κ: Neighbourhood size

procedure ESTIMATEQ($D_{\mathcal{N}_i^\kappa}$, K, T_c, κ)
  for $t = 1, \ldots, T_c + 1$ **do**
    Use $D_{\mathcal{N}_i^\kappa}$ to calculate

$$u_i^{K_k}(t) = \sum_{j \in \mathcal{N}_i^\kappa} [K_k]_{ij} x_j(t)$$

    Share $u_i^{K_k}(t)$ with the κ-neighbourhood.
    Use Equation (5.7)-(5.10) to form $\phi_i(t)$, $\psi_i(t)$, $c_i(t)$ and f_i.
  end for
  Estimate $\widehat{h}_i$ using Equation (5.11).
  Using Equation (5.12), transform $\widehat{h}_i$ into matrix form $\widehat{H}_i$.
  Share $\widehat{H}_{i12}$ and $\widehat{H}_{i22}$ with the κ-neighbourhood.
  By Equation (5.13), Agent i now knows $[\widehat{H}_{12}]_{ij}$ and $[\widehat{H}_{22}]_{ij}$ for all j.
  return $\widehat{H}_{11}, \widehat{H}_{12}, \widehat{H}_{22}$
end procedure

Even though Algorithm 1 and Algorithm 2 are written in terms of Agent i, all agents run these two algorithms. With the critic architecture in place, we now turn our attention to the actor architecture.

5.3 Actor Architecture: Decentralised Policy Update

In [14] the authors update their policy greedily which works well in the centralised case but require solving a linear system of equations. An alternative update rule is to update the policy in the direction of improvement using the gradient. We are considering deterministic policies and use the theory of deterministic policy gradients from [23]. The deterministic policy gradient is the expected gradient of the action-value function, which can be more efficiently estimated than the usual stochastic policy gradient [23]. Specifically, for a deterministic policy π and an objective function $J(\pi)$ we have

$$\nabla_\pi J(\pi) = \mathbb{E}\left[\nabla_\pi Q^\pi(x, \pi(x))\right] \tag{5.15}$$

Using the inner product definition of the gradient

$$\mathrm{d}Q(x,Kx) := \langle \nabla_K Q(x,Kx), \mathrm{d}K \rangle = \mathrm{tr}(\nabla_K Q(x,Kx)^\top \mathrm{d}K)$$

we get

$$\begin{aligned}
\mathrm{d}Q(x,Kx) &= 2x^\top H_{12}\mathrm{d}Kx + 2x^\top K^\top H_{22}\mathrm{d}Kx \\
&= \mathrm{tr}(2x^\top H_{12}\mathrm{d}Kx) + \mathrm{tr}(2x^\top K^\top H_{22}\mathrm{d}Kx) \\
&= \mathrm{tr}(2xx^\top H_{12}\mathrm{d}K) + \mathrm{tr}(2xx^\top K^\top H_{22}\mathrm{d}K) \\
&= \left\langle 2(H_{12}^\top + H_{22}K)xx^\top, \mathrm{d}K \right\rangle
\end{aligned}$$

and thus

$$\nabla_K Q(x,Kx) = 2(H_{12}^\top + H_{22}K)xx^\top \tag{5.16}$$

We note that the gradient depends on global information and set out to show that, in our setting, the partial derivative $\partial Q(x,Kx)/\partial [K]_{ij}$ can be approximated by Agent i only using local information. First of all, the matrices H_{12} and H_{22} are not known and have to be replaced with the estimates $\widehat{\underline{H}}_{12}$ and $\widehat{\underline{H}}_{22}$. Equation (5.14) tells us that these estimates are sparse, more specifically that $[\widehat{\underline{H}}_{12}]_{ij}, [\widehat{\underline{H}}_{22}]_{ij} = 0$ if $j \notin \mathscr{N}_i^{2\kappa-1}$. Combining this knowledge with the fact that $K \in \mathscr{K}^\kappa$, we get

$$[\widehat{\underline{H}}_{22}K]_{ij} = \sum_{l=1}^{N} [\widehat{\underline{H}}_{22}]_{il}[K]_{lj} = 0_{m_i \times n_j} \text{ if } j \notin \mathscr{N}_i^{3\kappa-2} \tag{5.17}$$

Using this in Equation (5.16) with the definition of partial derivative gives

$$\frac{\partial \widehat{Q}(x,Kx)}{\partial [K]_{ij}} = 2\sum_{l=1}^{N} [\widehat{\underline{H}}_{12}^\top]_{il}[xx^\top]_{lj} + 2\sum_{l=1}^{N} [\widehat{\underline{H}}_{22}K]_{il}[xx^\top]_{lj} \tag{5.18}$$

$$= 2\sum_{l \in \mathscr{N}_i^{2\kappa-1}} [\widehat{\underline{H}}_{12}^\top]_{il}[xx^\top]_{lj} + 2\sum_{l \in \mathscr{N}_i^{3\kappa-2}} [\widehat{\underline{H}}_{22}K]_{il}[xx^\top]_{lj} \tag{5.19}$$

From Equation (5.13), we see that in order to update Agent i's parameters using Equation (5.19) it is sufficient for Agent i to have access to $\widehat{\underline{H}}_{j12}$ and $\widehat{\underline{H}}_{j22}$ for all $j \in \mathscr{N}_i^\kappa$. Agent i also needs to calculate $[\widehat{\underline{H}}_{22}K]_{il}$ for $l \in \mathscr{N}_i^{3\kappa-2}$ and from Equation (5.14) and (5.17), we see that Agent i needs access to $[K]_{j:}$ for $j \in \mathscr{N}_i^{2\kappa-1}$. According to Equation (5.15), the final step in finding the gradient of the cost function is taking the expected value of the gradient in Equation (5.16). The actor achieves this by sampling a trajectory and estimating the mean of the gradient over this trajectory. The actor's procedure for Agent i is described in pseudocode in Algorithm 3. As mentioned for the critic architecture, Algorithm 3 is then run for each of the agents.

Algorithm 3 Actor: UPDATEK

Require:

$K_k \in \mathscr{K}^{\kappa}$: Current controller

$\widehat{H}_i$: Estimated Q-function parameters for all i

T_a: Trajectory length

α: Step size

κ: Neighbourhood size

1: **procedure** UPDATEK(K_k, $\widehat{H}_i$, T_a, α, κ)

2: Generate on-policy data $\{x_i^a(t)\}_{t=1}^{T_a}$ by following policy K_k.

3: Share $\{x_i^a(t)\}_{t=1}^{T_a}$ with the $(3\kappa-2)$-neighbourhood and share $[K_k]_{i:}$ with the $(2\kappa-1)$-neighbourhood.

4: Use $[\widehat{H}_{12}]_{ij}$ and $[\widehat{H}_{22}]_{ij}$ from ESTIMATEQ together with Equation (5.19) to estimate the partial derivative $G_{ij} := \partial J(K)/\partial [K_k]_{ij}$ using the standard estimator for expected value

$$\widehat{G}_{ij} := \frac{1}{T_a}\sum_{t=1}^{T_a} \frac{\partial \widehat{\underline{Q}}(x^a(t), K_k x^a(t))}{\partial [K_k]_{ij}}.$$

5: Update parameter $[K_k]_{ij}$ by taking a negative gradient step

$$[K_{k+1}]_{ij} = [K_k]_{ij} - \alpha \widehat{G}_{ij} \; \forall j \in \mathscr{N}_i^{\kappa}.$$

6: **return** K_{k+1}

7: **end procedure**

5.4 Scalable MARL for Network LQR

The critic and the actor are standalone procedures and we formulate Algorithm 4 for distributed learning of network LQR by combining Algorithm 1, 2 and 3.

Algorithm 4 Scalable MARL for Network LQR

Require:

κ: Neighbourhood size

$K_0 \in \mathscr{K}^\kappa$: initial stabilising controller

k_{max}: Number of policy iterations

T_c: Critic trajectory length

σ_η^2: Exploration variance

T_a: Actor trajectory length

α: Actor step size

$D_{\mathscr{N}_i^\kappa} \leftarrow$ CollectCriticData(K_0,T_c,σ_η^2, κ) for all i
for $k = 0, \ldots, k_{max} - 1$ **do**
  $\widehat{H}_i \leftarrow$ EstimateQ($D_{\mathscr{N}_i^\kappa}$, K_k, T_c, κ) for all i
  $K_{k+1} \leftarrow$ UpdateK(K_k,$\widehat{H}_i$, T_a, α, κ)
end for
return $K_{k_{max}}$

Algorithm 4 is a off-policy algorithm that returns a localised policy $K \in \mathscr{K}^\kappa$ after a set number of iterations specified by the user. It assumes access to an initial stabilising controller. For each policy update, the actor has to collect a trajectory of length T_a to estimate the expected value, it is thus not offline even though the critic part of the algorithm is designed to run in an offline manner. In practice, T_a can be small compared to T_c making the online data collection negligible.

Algorithm 4 also require several parameters that, in practice, need to be tuned to the problem at hand. Because of this, we have not investigated specific parameter choices, but instead leave that to the algorithm's implementer.

6

Simulation Study

This section evaluates the algorithms introduced in the previous section through a simulation study. First, Lemma 4.1 is visualised for a fabricated example satisfying the assumptions of the lemma. Then, the performance of Algorithm 4 is evaluated by applying it to the problem of controlling the temperature in a building.

6.1 Lemma 4.1 for a Toy Problem

We visualise Lemma 4.1 using a toy example. We let the graph be a chain of $N = 20$ agents as in Figure 3.1 and we take

$$L = \begin{pmatrix} .5 & .2 & & & \\ .2 & .5 & .2 & & \\ & \ddots & \ddots & \ddots & \\ & & .2 & .5 & .2 \\ & & & .2 & .5 \end{pmatrix}, \; [M_i]_{lj} = \begin{cases} 1 & \text{if } l = j \in \mathcal{N}_i \\ 0 & \text{otherwise} \end{cases} \tag{6.1}$$

L is banded, symmetric and stable with $\tau = 1$. Obviously it is also SED with for example, $c_L = 0.5$ and $\gamma = .9$. For M_i, we can take $c_{M_i} = e^{.9}$ and $\gamma = .9$ to make it (c_{M_i}, γ)-*SED away from i*. Using MATLAB's built-in functions we solve the Lyapunov equation from Lemma 4.1. The decay of P_i, as defined in Lemma 4.1, is visualised in Figure 6.1.

We continue to visualise a case when Lemma 4.1 fails to give an informative bound. Once again consider the graph topology given by Figure 3.1 with $N = 20$ agents and take

$$L = \begin{pmatrix} 0 & e & & & \\ & 0 & e & & \\ & \ddots & \ddots & \ddots & \\ & & & 0 & e \\ & & & & 0 \end{pmatrix} \tag{6.2}$$

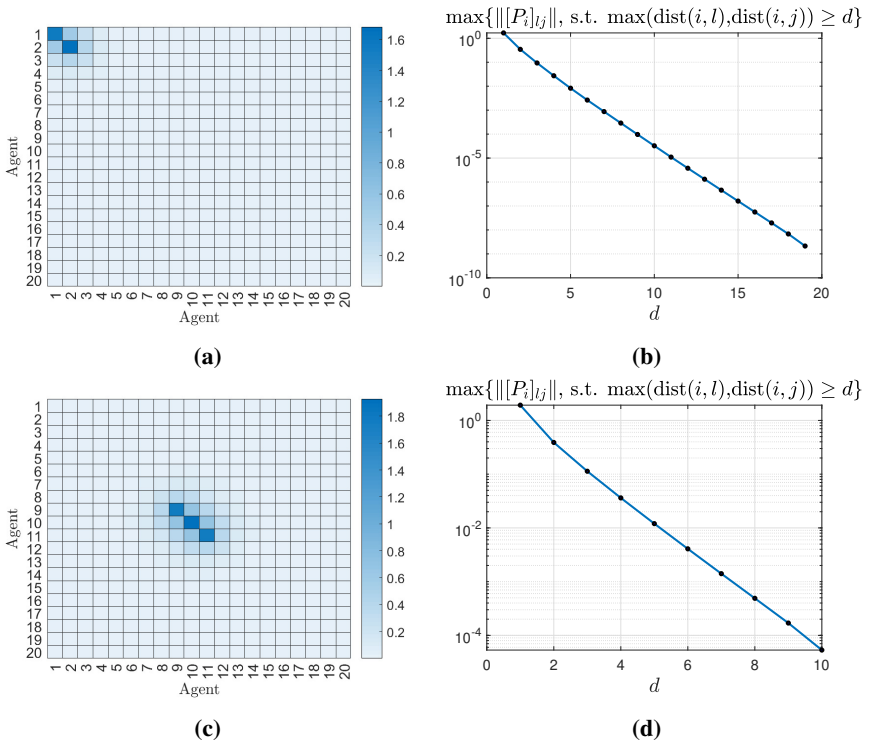


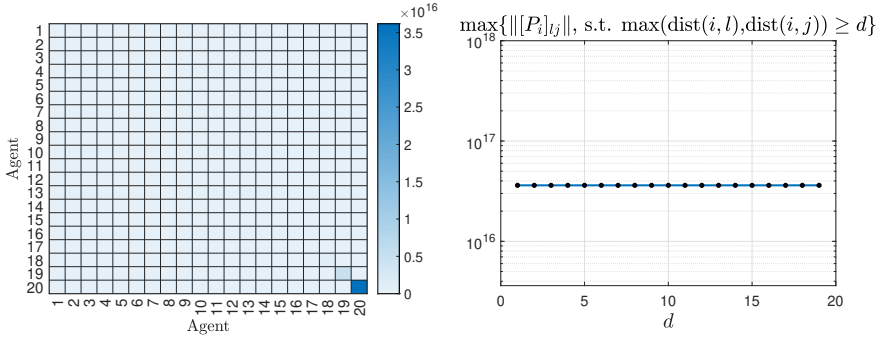
Figure 6.1 Decay of the solution P_i to the Lyapunov equation $P_i = L^\top P_i L + M_i$ with L, M_i specified by Equation (6.1). In (a) and (b) the decay is shown for $i = 1$. In (c) and (d) for $i = 10$.

Let M_i be as before, specified in Equation (6.1). L is still $(c_L, \gamma) - SED$ with $\gamma = .9$ and $c_L = e^2$. Moreover, L is stable as all its eigenvalues are zero. However, it is easy to see that

$$\max_k ||L^k|| = e^{N-1} = e^{19}$$

meaning $\tau > e^{19}$ in Definition 3.4. For this example, this means $c_{P_i} > e^{38} \approx 3 \cdot 10^{16}$ in Lemma 4.1 which is a non negligible constant. Figure 6.2 show that there is indeed no decay away from i with L as in Equation (6.2).

To give some intuition into why this example fails, we notice that L is a Jordan matrix with upper diagonal elements larger than 1. Every time L is multiplied by itself it's elements both grow and move towards the upper right corner, this pattern holds up until L^{20} which is the zero matrix and makes c_{P_i} huge for agents early in the chain. Before continuing we once again remind the readers that this is in fact not a counter example to Lemma 4.1, it is just an ill-conditioned example.



(a) **(b)**

Figure 6.2 Heatmap and decay plot of P_i with $i = 1$ when L is given by Equation (6.2) and M_i by Equation (6.1). Note the scale. This L does not break the assumptions of Lemma 4.1, nor does it give an informative bound on the decay rate of P_i.

6.2 Scalable MARL for Thermal Control

We consider the problem of controlling the temperature in a building, more specifically keeping the temperature at the desired steady state. The building is modelled as an undirected connected graph where each room has an agent (node in the graph) that controls the temperature in that room. If rooms i and j are adjacent the edge (i, j) exists. We consider a one-story building with $N = 25$ rooms arranged in a 5 by 5 square, see Figure 6.3.

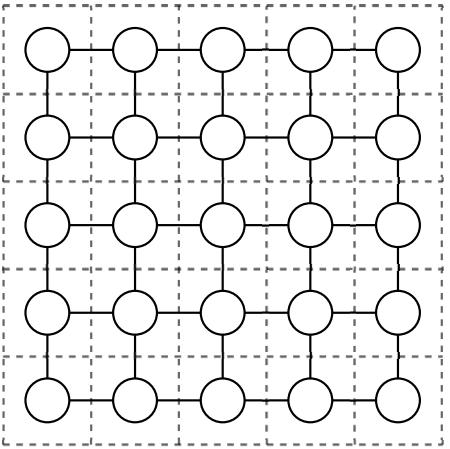


Figure 6.3 Schematic over the 25 room one-story building. Dashed lines represent walls in the building and solid lines the model's graph topology.

We assume the thermal dynamics model studied in [30] and [31] and add Gaus-

sian noise. In [30] they model the problem as a continuous-time LQR model where the state and control variables are expressed as deviations from the desired steady state temperature.

$$\min_{\{u(t)\}} \int_0^\infty \sum_{i=1}^{N} s_i x_i(t)^2 + u_i(t)^2$$
$$\dot{x}_i = \sum_{j \in \mathcal{N}_i, j \neq i} \frac{1}{v_i \zeta_{ij}} (x_j - x_i) + \frac{1}{v_i} u_i$$

Here v_i is the thermal capacitance of room i, ζ_{ij} the thermal resistance between two neighbouring rooms and s_i the relative cost of deviating from the desired temperature. We assume $\zeta_{ij} = 0.5°\text{C} / \text{kW}$, $v_i = 200 + 20 \times \mathcal{N}(0,1)$ kJ/°C and $s_i = 5$. Furthermore, we define the dynamics and and cost matrices for the continuous problem as

$$[A_c]_{ij} = \begin{cases} -\sum_{r \neq i, r \in \mathcal{N}_i} \frac{1}{v_i \zeta_{ij}} & \text{if } i = j \\ \frac{1}{v_i \zeta_{ij}} & \text{if } i \neq j, j \in \mathcal{N}_i \\ 0 & \text{otherwise} \end{cases},$$
$$B_c = \text{diag}(\frac{1}{v_i}),\ S_c = \text{diag}(s_i),\ R_c = I$$

Using $\Delta t = 1/4$ we discretise the system using the scheme in [30] yielding the discrete-time LQR matrices A, B, R and S.

$$A = e^{\Delta t A_c},\ B = \left(\int_0^{\Delta t} e^{s A_c} ds \right) B_c,\ S = \Delta t S_c,\ R = \Delta t R_c$$

Before testing our algorithms, the SED assumption on A and B is verified in Figure 6.4. We test the critic architecture by running Algorithm 1 and 2 for different trajectory lengths T_c. We set $K = K_{play} = -I$, system noise $w(t) \sim \mathcal{N}(0,1)$ and run 10 trials. In Figure 6.5 the relative error for H and its submatrices H_{11}, H_{12} and H_{22} is shown for different values on T_c and constant $\sigma_\eta = 10$.

We now evaluate Algorithm 4 by comparing its output to the optimal controller. First the optimal controller is calculated with Equation (2.1) and then our algorithm is applied to problem (P) with system noise $w(t) \sim \mathcal{N}(0,1)$. We initialise the algorithm with $K_0 = -3I$, $T_a = 10000$, exploration noise $\sigma_\eta = 10$ and actor step size $\alpha = 0.005$. We first run the algorithm using the real, but truncated action-value function for $\kappa = 1, 3, 5$ which was calculated using Equation (2.7) and then truncated using Definition 4.1. The full algorithm was then run with $T_c = 30000$ and $T_c = 100000$. The relative cost $J(K)$ for different values on κ can be seen in Figure 6.6.

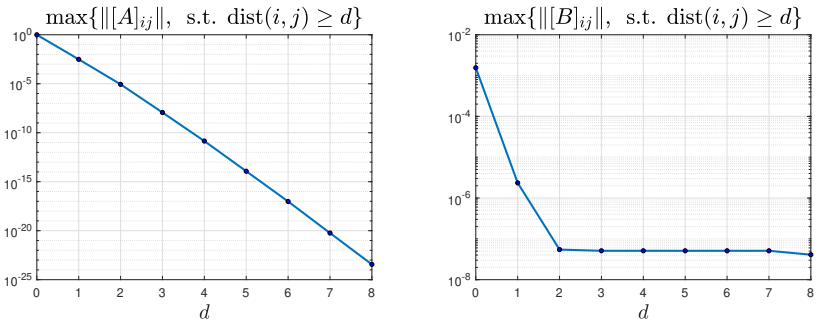


Figure 6.4 Decay of system matrices A and B in Example 6.2. In (a) the maximum norm of the submatrix $[A]_{ij}$ is seen when the distance between i and j is at least d varying from 0 to 8 which is the maximum distance between any agents in the graph. Similarly in (b) for the B matrix.

We continue by studying a similar system but with faster dynamics. Keeping everything else as before, we run the same experiments on a system with v_i scaled down by a factor of 10 such that, $v_i = 20 + 2 \times \mathcal{N}(0,1)$ kJ/°C. Both A_c and B_c are inversely proportional to v_i and thus decreasing this parameter has the effect of increasing the speed of the dynamics. The critic's performance for this system can be seen in Figure 6.7 and the relative cost achieved by running Algorithm 4 can be seen in Figure 6.8. In the next section, we proceed to discuss our findings further.

6.3 Results and Discussion

Section 6.1 illustrates how Lemma 4.1 can be used to ensure decay in Figure 6.1 and the importance of checking parameters' dependency on N in Figure 6.2 for chained agents. However, this experiment also visualises that even if L is SED and stable, if the stability constant τ has a strong dependence on N, it is not certain Lemma 4.1 provides a useful bound. When applying Lemma 4.1 to a system it is therefore important to verify that τ in Definition 3.4 for the L matrix does not have a exponential dependence on N, that is verifying that L^k does not grow exponentially with the size of the system. For normal matrices, the spectral norm coincides with the maximum eigenvalue of the matrix. Therefore, in the case when L is normal and stable, Defintion 3.4 holds with $\tau = 1$, and thus no further calculations are needed to check dependence on N.

More importantly, in Section 6.2 both the critic and the full actor-critic framework are evaluated on a thermal control problem. First the critic was evaluated and as can be seen in Figure 6.5, the relative error in the parameters of the Q-function decreases with longer trajectory length, except for the $\kappa = 1$ case. The smallest rel-

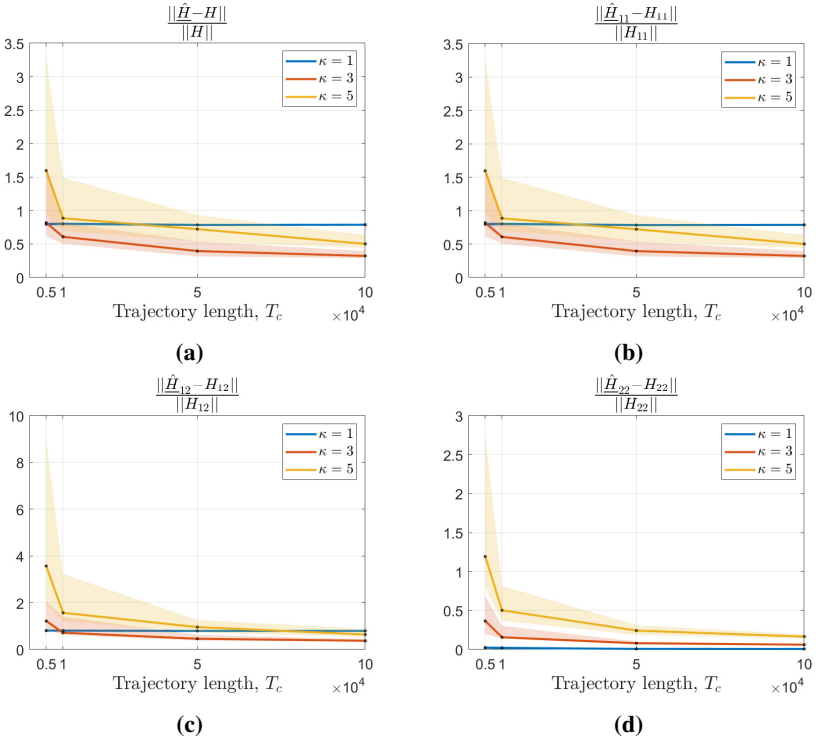


Figure 6.5 Median relative error for the estimated submatrices $\widehat{H}_{11}$, $\widehat{H}_{12}$ and $\widehat{H}_{22}$ shown by the solid lines. The shaded regions represent 90 % confidence intervals of the estimate.

ative error is achieved for the estimate of H_{22}, which is due to the scaling of the problem. In Figure 6.6, the relative cost is shown and the lowest cost is achieved for $\kappa = 3$ for both values of the trajectory length T_c. With $T_c = 100000$ and $\kappa = 3$, Algorithm 4 finds a distributed controller with a cost that is around 2 % higher than the optimal. From this figure one can also see that a larger κ does not mean lower cost, and thus, if possible, κ should be fine tuned to the problem. In theory, a larger κ should never increase the cost as irrelevant parameters can be set to zero. However, in practice more parameters can decrease the performance as was shown in [14] for the centralised critic case. This is also clear from Figure 6.6 (a) where the real, truncated, action-value function was used. This figure also visualizes the part of the error due to truncation. If we now combine Figure 6.5 and Figure 6.6 it is clear that our algorithm can be robust to bad critic estimates.

When considering the system with smaller v_i, both the critic and the full algorithm perform better compared to the case with larger v_i as shown in Figure 6.7 and 6.8. As previously discussed a smaller v_i means faster dynamics which intuitively means fewer samples are needed as the state space is explored faster. How-

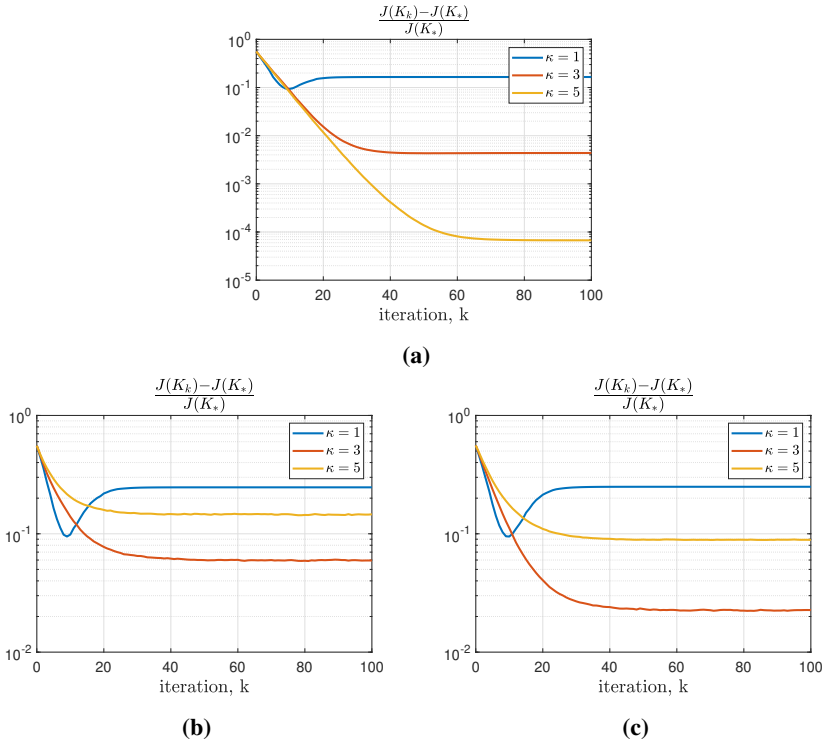


Figure 6.6 Relative cost of learned thermal controllers. In figure (a), the real, but truncated, Q-function was used. In (b) and (c) the Q function was also learned using $T_c = 30000$ in (b) and $T_c = 100000$ in (c).

ever, it is worth noting that, in reality v_i is a physical parameter which cannot be chosen freely. Interestingly the error for $\kappa = 1$ does not decrease suggesting the error in this case is due to truncation and not due to too few samples. On the other extreme with $\kappa = 5$ the critic's performance is better than for other values of κ if enough samples are used. The error is also much smaller compared to the slower system in Figure 6.5 for both $\kappa = 3$ and $\kappa = 5$. Considering the relative cost, in the case of $T_c = 100000$ and $\kappa = 5$, after 100 iterations our algorithm finds a controller with a 0.3 % higher cost than the optimal controller.

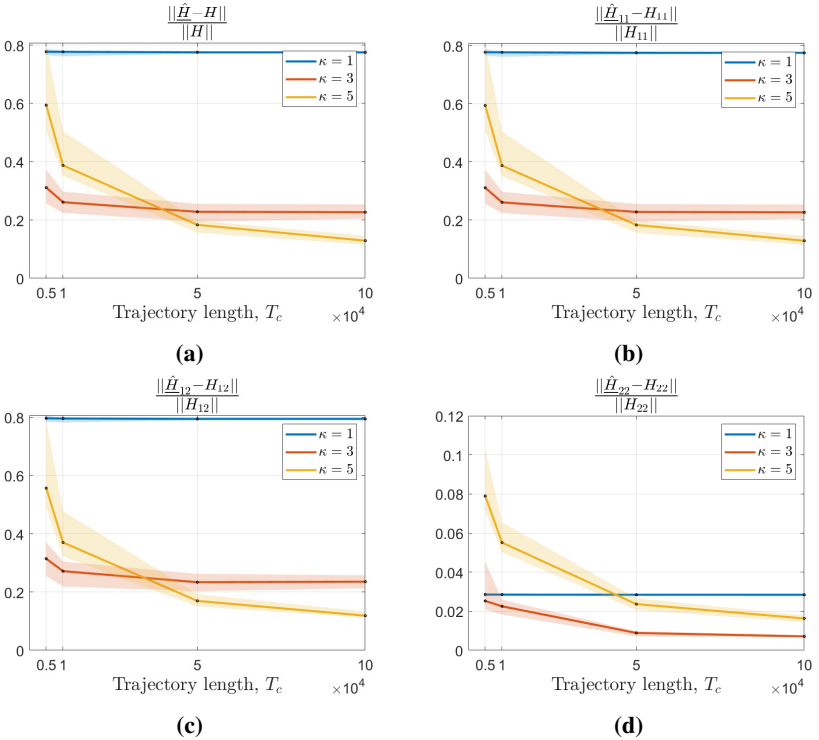


Figure 6.7 Median relative error for the estimated submatrices $\underline{\widehat{H}}_{11}$, $\underline{\widehat{H}}_{12}$ and $\underline{\widehat{H}}_{22}$ shown by the solid lines. The shaded regions represent 90 % confidence intervals of the estimate. In this example the thermal capacitance for room i was scaled down by a factor of 10 compared to the one used in Figure 6.5.

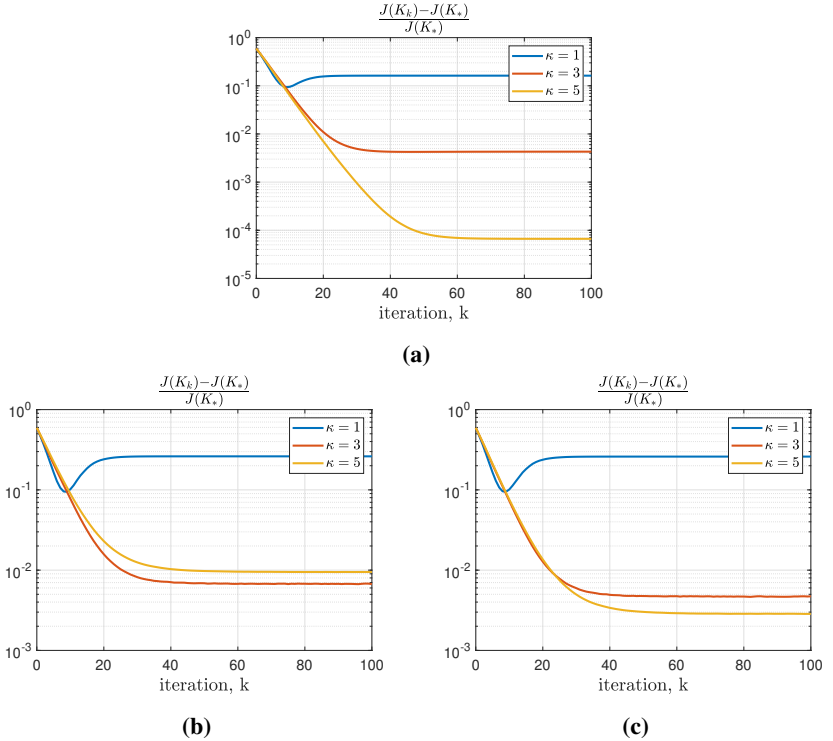


Figure 6.8 Relative cost of learned thermal controllers. In figure (a), the real, but truncated, Q-function was used. In (b) and (c) the Q function was also learned using $T_c = 30000$ in (b) and $T_c = 100000$ in (c). Compared to Figure 6.6 the thermal capacitance for room i was scaled down by a factor of 10.

7

Conclusions and Future Work

7.1 Conclusions

This thesis extends previous work on the structure of the optimal controller for networked LQR with spatially decaying structure, to the problem of using samples to learn a distributed controller. First, the individual agents' value and action-value functions were defined and their structure investigated. It was shown that, it is indeed possible to bound the error when truncating these functions. These results were then used to design a scalable reinforcement learning algorithm for finding distributed controllers for the network LQR problem. Finally, the algorithm's performance was evaluated through a simulation study. These simulations show, that although the algorithm requires a lot of samples, near optimal cost can be achieved.

7.2 Limitations and Directions for Future Work

Below we list the main limitations of Algorithm 4 and suggestions on how to handle these. Furthermore, future directions which we believe to be fruitful are discussed.

First of all in Lemma 4.1 the decay parameter γ_{P_i}, always depends on N and similar to the discussion in [30] it would be interesting to see if this dependence is fundamental or a consequence of the proof. By carefully going through the proof of Lemma 4.1, one can see that the provided bound is not tight. In this work we prioritized simplicity over tightness and thus leave tightening the bounds to future work. The results in Section 4 could also potentially be extended to the continuous-time setting and the case of an infinite number of agents ($N \to \infty$).

If we instead consider Algorithm 4, one of the most significant drawbacks is that it is sample-inefficient. Although, this work has focused on spatial locality and performance loss due to truncation, sample efficiency is important in the real world. If it is possible to simulate the system, the number of samples might not be a problem. However, it is possible that model-based methods can outperform our algorithm in

this case. Sample inefficiency is typical for reinforcement learning algorithms [29] and a different approach would probably be needed in order to avoid this problem.

In its current form the algorithm is also offline in the sense that we first collect data for the critic and then use the same data throughout the algorithm. By small adjustments to the algorithm, an online version could be implemented which might be more beneficial if it is possible to continuously collect data from the system. An online version of Algorithm 4 would however require the collection of T_c samples for each policy iteration which might be unreasonable if T_c is large. The actor currently collects T_a samples each iteration in order to estimate an expected value. With the current actor architecture, this is needed as the actor needs access to on-policy samples in order to estimate the gradient.

To avoid resampling each iteration, a greedy actor could be used instead. In the centralised case, a greedy actor updates the policy by solving a linear system of equations. In [18] they propose a technique to solve sparse linear systems distributively and it would be interesting to investigate if their techniques could be used to implement a scalable greedy actor. Designing and evaluating such an architecture against our gradient based actor is an interesting future direction.

Another limitation of Algorithm 4 is its dependence on parameters such as neighbourhood size κ and step size α. These require fine tuning to achieve good performance which might mean trial and error based search. Similarly, the system assumptions can be hard to validate in practice and, as has been shown, this can lead to poor performance.

In this work we have also not considered effects of measurement errors and effects of long-range-correlated noise. These will almost surely affect the performance of Algorithm 4 and should be considered when implementing the algorithm.

Ignoring the algorithm's limitations, another important direction of future work is analysis of the algorithm. Such analysis could potentially lead to guarantees regarding performance and sampling complexity. Performance analysis is, at the time of writing, being investigated by initially ignoring errors stemming from sampling. On a high level, the idea is to first bound the estimation error for the Q-function. Step two would then be to prove gradient descent convergence, at least a stationary point, when using the approximated Q-function for estimating the gradient. A first approach would to this problem would be to try and use techniques from the similar work in [17]. Hopefully these results could then be extended to show convergence to local minima. Finally, it would be interesting to investigate the algorithm's performance in practice, by applying it to other problems in network learning and control.

Bibliography

[1] Y. Abbasi-Yadkori, N. Lazic, and C. Szepesvári. "Model-free linear quadratic control via reduction to expert prediction". In: *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR. 2019, pp. 3108–3117.

[2] M. Andreasson. *Control of multi-agent systems with applications to distributed frequency control power systems*. PhD thesis. KTH Royal Institute of Technology, 2013.

[3] K. J. Åström and B. Wittenmark. *Computer-controlled systems: theory and design*. Courier Corporation, 2013.

[4] D. P. Bertsekas. *Dynamic Programming and Optimal Control*. 3rd. Vol. II. Athena Scientific, Belmont, MA, USA, 2005.

[5] S. J. Bradtke, B. E. Ydstie, and A. G. Barto. "Adaptive linear quadratic control using policy iteration". In: *Proceedings of 1994 American Control Conference-ACC'94*. Vol. 3. IEEE. 1994, pp. 3475–3479.

[6] L. Buşoniu, R. Babuška, and B. De Schutter. "Multi-agent reinforcement learning: An overview". *Innovations in multi-agent systems and applications-1* (2010), pp. 183–221.

[7] M. Fazel, R. Ge, S. Kakade, and M. Mesbahi. "Global convergence of policy gradient methods for the linear quadratic regulator". In: *International conference on machine learning*. PMLR. 2018, pp. 1467–1476.

[8] H. Feng and J. Lavaei. "On the Exponential Number of Connected Components for the Feasible Set of Optimal Decentralized Control Problems". In: *2019 American Control Conference (ACC)*. 2019, pp. 1430–1437. DOI: 10.23919/ACC.2019.8814952.

[9] D. Görges. "Distributed adaptive linear quadratic control using distributed reinforcement learning". *IFAC-PapersOnLine* **52**:11 (2019), pp. 218–223.

[10] R. A. Horn and C. R. Johnson. *Matrix analysis*. Cambridge university press, 2012.

[11] J. Ibarz, J. Tan, C. Finn, M. Kalakrishnan, P. Pastor, and S. Levine. "How to train your robot with deep reinforcement learning: lessons we have learned". *The International Journal of Robotics Research* **40**:4-5 (2021), pp. 698–721.

[12] J. Klamka. "Controllability of linear dynamical systems". *Contrib. Theory Differ. Equ* **1** (1963), pp. 189–213.

[13] V. Konda and J. Tsitsiklis. "Actor-critic algorithms". *Advances in neural information processing systems* **12** (1999).

[14] K. Krauth, S. Tu, and B. Recht. "Finite-time analysis of approximate policy iteration for the linear quadratic regulator". *Advances in Neural Information Processing Systems* **32** (2019).

[15] M. G. Lagoudakis and R. Parr. "Least-squares policy iteration". *The Journal of Machine Learning Research* **4** (2003), pp. 1107–1149.

[16] D. Li, D. Zhao, Q. Zhang, and Y. Chen. "Reinforcement learning and deep learning based lateral control for autonomous driving [application notes]". *IEEE Computational Intelligence Magazine* **14**:2 (2019), pp. 83–98.

[17] Y. Li, Y. Tang, R. Zhang, and N. Li. "Distributed reinforcement learning for decentralized linear quadratic control: A derivative-free policy optimization approach". *IEEE Transactions on Automatic Control* **67**:12 (2021), pp. 6429–6444.

[18] A. Minot, Y. M. Lu, and N. Li. "A distributed Gauss-Newton method for power system state estimation". *IEEE Transactions on Power Systems* **31**:5 (2015), pp. 3804–3815.

[19] G. Qu, Y. Lin, A. Wierman, and N. Li. "Scalable multi-agent reinforcement learning for networked systems with average reward". *Advances in Neural Information Processing Systems* **33** (2020), pp. 2074–2086.

[20] G. Qu, A. Wierman, and N. Li. "Scalable reinforcement learning for multiagent networked systems". *Operations Research* **70**:6 (2022), pp. 3601–3628.

[21] B. Recht. "A tour of reinforcement learning: The view from continuous control". *Annual Review of Control, Robotics, and Autonomous Systems* **2** (2019), pp. 253–279.

[22] Y. Shoham and K. Leyton-Brown. *Multiagent systems: Algorithmic, game-theoretic, and logical foundations*. Cambridge University Press, 2008.

[23] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. "Deterministic policy gradient algorithms". In: *International conference on machine learning*. Pmlr. 2014, pp. 387–395.

[24] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al. "Mastering the game of go without human knowledge". *nature* **550**:7676 (2017), pp. 354–359.

[25] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. 2018.

[26] A. Tsiamis, I. Ziemann, N. Matni, and G. J. Pappas. "Statistical learning theory for control: A finite sample perspective". *arXiv preprint arXiv:2209.05423* (2022).

[27] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, et al. "Grandmaster level in StarCraft II using multi-agent reinforcement learning". *Nature* **575**:7782 (2019), pp. 350–354.

[28] H. S. Witsenhausen. "A counterexample in stochastic optimum control". *SIAM Journal on Control* **6**:1 (1968), pp. 131–147.

[29] Y. Yu. "Towards Sample Efficient Reinforcement Learning." In: *IJCAI*. 2018, pp. 5739–5743.

[30] R. Zhang, W. Li, and N. Li. "On the Optimal Control of Network LQR with Spatially-Exponential Decaying Structure". *arXiv preprint arXiv:2209.14376* (2022).

[31] X. Zhang, W. Shi, X. Li, B. Yan, A. Malkawi, and N. Li. "Decentralized temperature control via HVAC systems in energy efficient buildings: An approximate solution procedure". In: *2016 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. IEEE. 2016, pp. 936–940.

[32] Y. Zhang, G. Qu, P. Xu, Y. Lin, Z. Chen, and A. Wierman. "Global convergence of localized policy iteration in networked multi-agent reinforcement learning". *Proceedings of the ACM on Measurement and Analysis of Computing Systems* **7**:1 (2023), pp. 1–51.

Lund University
Department of Automatic Control
Box 118
SE-221 00 Lund Sweden

Document name
MASTER'S THESIS

Date of issue
September 2023

Document Number
TFRT-6207

Author(s)
Johan Olsson

Supervisor
Na Li, Harvard University, USA
Emma Tegling, Dept. of Automatic Control, Lund University, Sweden
Anders Rantzer, Dept. of Automatic Control, Lund University, Sweden (examiner)

Title and subtitle
Scalable Reinforcement Learning for Linear-Quadratic Control of Networks

Abstract

Distributed optimal control is known to be challenging and can become intractable even for linear-quadratic regulator problems. In this work, we study a special class of such problems where distributed state feedback controllers can give near-optimal performance. More specifically, we consider networked linear-quadratic controllers with decoupled costs and spatially exponentially decaying dynamics. We aim to exploit the structure in the problem to design a scalable reinforcement learning algorithm for learning a distributed controller. Recent work has shown that the optimal controller can be well approximated only using information from a κ- neighbourhood of each agent. Motivated by these results, we show that similar results hold for the agents' individual value and action-value functions. We continue by designing an algorithm, based on the actor-critic framework, to learn distributed controllers only using local information. Specifically, the action-value function is estimated by modifying the Least Squares Temporal Difference for Q-functions method to only use local information. The algorithm then updates the policy using gradient descent. Finally, the algorithm is evaluated through simulations which suggest near-optimal performance.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280-5316

ISBN

Language
English

Number of pages
1-55

Recipient's notes

Security classification